2005

Knowledge-based fault detection using time-frequency analysis

Venkata S. Vongala
Louisiana State University and Agricultural and Mechanical College, vvonga1@lsu.edu

KNOWLEDGE-BASED FAULT DETECTION USING TIME-FREQUENCY ANALYSIS

A Thesis

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering

in

The Department of Electrical and Computer Engineering

by
Venkata S Vongala
Bachelor of Engineering, Andhra University, April 2003
December 2005

Acknowledgments

First and foremost, I express my sincere appreciation and thanks to my advisor and major professor, Dr. Jorge Aravena, for his constant guidance and valuable comments. I sincerely thank Dr. Kemin Zhou and all my peers working on the project for their valuable suggestions. Their constant encouragement and discussions have helped me gain deep insight in this work. I also express my gratitude to Dr. Guoxiang Gu and Dr. Bahadir Gunturk for serving in my defense committee.

I thank the members of my family for always being there to inspire, encourage, and support me. Last but certainly not the least, I thank my friends here at LSU, whose company and friendship I dearly cherish. Thank you for making my stay here at LSU a memorable one.

Table of Contents

List of Tables

List of Figures

Abstract

This work studies a fault detection method which analyzes sensor data for changes in their characteristics to detect the occurrence of faults in a dynamic system. The test system considered in this research is a Boeing-747 aircraft system and the faults considered are the actuator faults in the aircraft. The method is an alternative to conventional fault detection method and does not rely on analytical mathematical models but acquires knowledge about the system through experiments.

In this work, we test the concept that the energy distribution of resolution than the windowed Fourier transform.

Verification of the proposed methodology is carried in two parts. The first set of experiments considers entire data as a single window. Results show that the method effectively classifies the indicators by more that 85% as correct detections. The second set of experiments verifies the method for online fault detection. It is observed that the mean detection delay was less than 8 seconds. We also developed a simple graphical user interface to run the online fault detection.

Chapter 1
Introduction

1.1 Fault Detection and Isolation

Fault Detection and Isolation (FDI) is important from the view point of improving system availability and reliability. A fault is here to be understood as any kind of malfunction in actual dynamic systems that lead to an anomaly in overall system performance. Such malfunctions may occur due to component failures, actuator failures or instrument failures in the sensors. Any anomaly in a system operation should be discovered at an early stage and the cause should be figured out. The diagnosis of the damaged component should be performed at the time when faults are still small and their effect is not yet harmful to the process. Timely detection of the abnormalities prevents extensive damage to persons, the environment and the system themselves. This issue has been one of the primary concerns in aircraft safety system, where progressive faults in the actuators not detected in early stages can lead to disaster. This research focuses on early detection of such faults and is a part of the project, "Aircraft Safety : Control Upset Management", sponsored by NASA and Louisiana Board of Regeants under the ESPCOR 2000 program.

The concept of fault diagnosis can be regarded as a three step algorithm. First of all, one or several signals are generated which reflect faults in the process behavior. These signals, generally called indicators, are composed from existing measurements and sometimes corresponding reference signals. The second step is the fault detection. In this step the indicators are evaluated to determine the presence of faulty behavior and a decision has to be made determining the time and location of the possible faults from the indicators. Finally, in the last step, the nature and

the cause of the fault is studied by analyzing the relations between the symptoms and determine which component of the system failed to operate. This corresponds to fault isolation.

In order to discover the fault in process behavior one should have some additional information which could either be an additional hardware (multiple sensors measuring the same quantity) or could be expressed analytically by the use of mathematical descriptions (process models) and by making use of statistical properties obtained from the history of process operation records respectively.

For the last two decades, there have been extensive research efforts on developing model-based FDI techniques [25],[6],[24],[8]. A residual signal is generated by comparing the measured output signal and the estimated one from a nominal system model. After being processed, this residual can be used as the indicator of abnormal behavior (faults) of the system. Model-based techniques rely on analytical redundancy. Analytically generated "measurement" outputs are compared with hardware measurements by using present and/or previous values of some variables in conjunction with their mathematical relationships. The fault detection process as discussed above, consists of three major tasks: (1) residual generation that entails taking the difference between the analytical and measured values; (2)statistical testing and signature generation; (3)diagnostics and decision making.

Generally speaking, it is difficult to design model-based FDI for nonlinear or uncertain systems. To tackle this problem, knowledge based methods have been proposed and studied [8]. The research in this area became more vigorous recently due to the machine intelligence-based methods. Without the need for a complete analytical model, these methods rely on the data-driven and knowledge based techniques to estimate the system dynamics. In this case, heuristic knowledge from the training processes is of great importance. Some of the knowledge- based methods

also use certain models built by Neural Networks [33], fuzzy system or expert systems, for mapping the inputs and outputs of the unknown system. Residual signals are then generated to detect and locate the faults in a similar fashion as in model based methods. In many other techniques, different operating conditions including normal and abnormal ones are treated as patterns, then a particular scheme is applied to analyze the online measurement data and map them to a known pattern directly so that the current system condition is identified. A variety of signal processing techniques have also been applied to help detect the faults/failures, such as spectrogram, the Wigner Ville distribution and wavelet decomposition attributed to their excellent time-frequency analysis properties. One major advantage of these 'model free', direct FDI techniques is their superior capability in identifying/classifying the faults. However, the inevitable noise, disturbances and uncertainties that are not accounted in the training processes usually make the online feature clusters differ from the expected ones in the built in feature table, causing errors in fault diagnosis.

This research is focussed on the practical situation where reliable models are not available. In such situations, one must use only the output data supplied by sensors. Our contention is that one can use signal processing techniques alone on the sensor data and enhance the effect of the fault, i.e make it more apparent to the operator for the purpose of fault detection. This research analyzes system's data for any change in the characteristics from the pre-fault to post-fault in order to detect the occurrence of faults.

Changes in the signal due to faults introduce distinctive and detectable changes in the energy distribution of the data. To describe the energy distribution, we chose continuous wavelet transform which analyzes a signal in the time-frequency domain. The analysis is carried out in two phases: Training phase and Classification

phase. In the training phase, instead of using mathematical models to describe the variations that could be expected, we use empirical observation approach analyzing extensive data sets where the performance of the plane is known. A sensitivity analysis is performed on the data to extract only the important sensor data and also a sensitivity analysis is performed to obtain the most sensitive scales. Using these scales the continuous wavelet transform is applied to obtain the indicators. Later using the indicators we define signatures characterizing the different behaviors. The actual fault detection is performed as a classification phase. Nearest neighbor data clustering technique is used to classify real time data into correct detections. The classification phase includes analyzing data coming in real time and determining the corresponding indicators. The distance of these indicators to the trained signatures are calculated and the indicators are grouped to the cluster represented by the closest signature subspace.

This research is an approach in which distinct analysis tools are integrated into a unified procedure:

- Time-frequency analysis using wavelets.
- Sensitivity analysis .
- SVD techniques for subspace detection.
- Data clustering .

New experimental data from a detailed B747 aircraft model have been used to test the proposed schemes. The test system is a closed loop non-linear model obtained from software package FTLAB747 simulated using Matlab's Simulink. A variety of faults have been simulated and introduced to study the performance of proposed fault detection methods. The different faults considered are the failure

in each of the four control surfaces: elevator, aileron, rudder, stabilizer and in the engine.

1.2 Outline of Thesis

The thesis is organized as follows.

In chapter 2,we give some background and state the previous works in fault detection, feature extraction, principal component analysis and clustering techniques. A special attention is given to wavelet based time-frequency analysis and feature extraction.

In chapter 3, we provide the necessary technical background for time-frequency analysis. This chapter explains the need for time-frequency analysis. We then discuss short time fourier transform and wavelet transform and later introduce the key concepts of scalogram.

In chapter 4, we describe our proposed approach in brief. Also in this chapter we explain the simulator used to collect data for our analysis.

In chapter 5, we describe the implementation of our methodology. First, we discuss the training of spectral signatures , followed by the classification of unknown signatures. Second, the strategy for real-time fault detection is given. Third, an attempt for fault isolation is discussed.

In chapter 6, we verify the proposed approach by testing its performance for data obtained from the B747 aircraft model. Experiments are carried to evaluate the performance and the obtained results are presented along with discussions.

We give a conclusion and summarize future work in chapter 7.

Chapter 2
Literature Review

2.1 Time-Frequency Representation

The earliest form of function representation using orthogonal basis functions is undoubtedly the Fourier series for continuous and periodic signals. However, one of the major drawbacks is that, the magnitude of Fourier Transform only provides spectral content with no indication about the time localization of the spectral components. Therefore, the analysis of non-stationary signals, whose spectral content change with time, requires a time-frequency representation (TFR), rather than just a frequency representation.

The first modification to the Fourier transform, to allow analysis of non-stationary signals came as short time Fourier transform (STFT). The idea behind the STFT was segmenting the signal by using a time-localized window, and performing the analysis for each segment. Since the Fourier transform was computed for every windowed segment of the signal, STFT was able to provide a true time-frequency representation. Dennis Gabor, who was interested in representing a communication signal using oscillatory basis functions in a time frequency plane, was the first one to modify the Fourier transform into STFT in 1946. Shortly after, in 1947, Jean Ville devised a similar TFR for representing the energy of a signal in the time-frequency plane (the Wigner-Ville transform). Many other TFRs have been developed between late 1940s and early 1970s, each of which differed from the other ones only by the selection of the windowing function.

However, all these TFRs suffered from one main drawback: they all used the same window for the analysis of the entire signal. In late 1970s, J.Morlet was faced with

the problem of analyzing signals which had very high frequency components with short time spans, and low frequency components with long time spans. He came up with the ingenuous idea of using a different window function for analyzing different frequency bands making advent of wavelets as a time-frequency representation of a signal.

Time-frequency analysis provides a powerful tool for the analysis of non-stationary signals [20]. Applications have been demonstrated in music signal analysis [32], machinery diagnosis [26], fault detection [11, 27], power engineering applications [40], damage detection [41], seismic monitoring [22], medical signal analysis [42], and acoustic and speech processing [30]. This research emphasizes the use of wavelet based time-frequency analysis in fault detection and isolation.

2.2 Wavelet Based Time-Frequency Analysis

The theory of wavelets originates from several different sources overlapping each other. Contrary to common beliefs, however, wavelets have a quite long and fascinating history in mathematics. This false opinion is probably due to the fact the word "wavelet" does not appear in literature until the 1980s when applications in signal and image processing started to emerge. In 1910 , the mathematician Alford Haar was the first to produce a complete orthonormal set for the Hilbert space $L^2(R)$. The orthonormal set is, in a sense, the building block of wavelet theory. The interest in the field activated only during early 1980's, beginning with the work of J.Morlet(1982). The results obtained by him though encouraging were not well received by the mathematical community. It was A.Grossman(1984) who laid a firm foundation to the theory. His work besides gaining mathematical respectability triggered active research in the field. It was however not until the work of Stehane Mallat and Yves Meyer in the late 1980's that wavelets entered mainstream science. By combining signal processing theory of quadrature mirror filters

and orthonormal wavelet basis, Mallat came up with the concept of "multiresolution analysis". After Mallat's contribution one more major breakthrough took place: in 1998 Ingrid Daubechies constructed wavelets with a preassigned degree of smoothness which made applied work much easier.

Mallat was the first to suggest the multi-resolution analysis via wavelet transformation. Its adequacy in application to fault detection and localization was further confirmed by other authors (Al-Rawi and Devaney, 1998 , Aravena and Chowdhury, 1996, Brito *et al.,* 1998, Galli and Heydt, 1995, 1997, and Huang *et al.,* 1997),

Robertson et al. (1994, 1996) used non-orthogonal wavelet as a pre-processor to investigate the feature of power system abnormalities. According to Brenner and Groutage, 2001 nonstationary time-frequency analysis is used for identification and classification of system dynamics. Their method is based on multiresolution signal decomposition (Mallat, 1989) which uses the concept that the mother wavelet is more localized in time at lower scales than at higher scales. This is the reason for fast and short transient disturbances being detected at lower scales and long transient disturbances being detected at higher scales. In this research we use the continuous wavelet transform for performing the time-frequency analysis.

2.3 Sensitivity Analysis

Sensitivity analysis can be viewed as a preprocessing step which removes distracting variance from a dataset so that downstream classifiers perform better. Sensitivity analysis should be applied, wherever appropriate, to keep only the important information while discarding noise and removing correlations. Sensitivity analysis methods are based upon different measures of sensitivity. In this research we are interested in obtaining the sensors that are most sensitive to a fault and also a second sensitivity analysis for selection of scales that show sensitivity to various faults.

There have been several efforts to develop sensitivity analysis mechanisms for time-series data . In [38], Discrete Fourier Transform(DFT) was employed to map time-series data from the time domain to the frequency domain. After dropping all but the first few frequency coefficients (which represent the lower frequencies of the signal), represent the approximate time-series data. Wu et al(2003) first proposed to use the Wavelet Transform to replace the well-accepted DFT for various reasons; the computation of WT is more efficient than DFT in general. A much better technique compared to [38], was proposed by Wu et al., which proposes to use the k-smallest coefficients of each time series, because they preserve the optimal amount of energy per time series. Also Morchen(2003), instead of using only first few coefficients of a wavelet transform, proposed a new method of choosing the best scales for a set of time series by evaluating a criterion function.

In this research we are not concerned at obtaining an approximation of the signal, rather we look at getting the disturbances present in the signal. These disturbances may correspond to high frequencies which in turn are the first few scales for the wavelet transform. In this work we propose a method in which use of all the scales that are important and sensitive to fault is made, instead of using only the first few scales. The approach captures the frequencies at which there is noticeable difference in normal and abnormal data. From these frequencies, corresponding scales for wavelet transform are computed. Compared to choosing the first few scales, this approach offers more semantic power. By using arbitrary scales, not only high frequencies but also any frequency can be included, if it is of importance in the data set.

2.4 Principal Component Analysis

By projecting the data into a lower-dimensional space that accurately characterizes the state of the process, dimensionality reduction techniques can greatly simplify

and improve process monitoring procedures. Principal Component Analysis (PCA) is a dimensionality reduction tool for monitoring industrial processes and has been studied by several academic and industrial engineers.

Kosanovich and Piovoso (1992,1994) made noted contributions in applying PCA to plant data at DuPont and other companies. Later Raich and Cinar(1995,1996) performed similar studies based on data collected from computer simulations of processes. Storer and Georgakis [24] proposed a technique of fault detection and isolation by using dynamic PCA.

PCA has been studied in papers of : Bakshi and Stephenoponlos, 1996; Ku et.al.,1995; MacGregor et.al.,1994 ; Pearson and Ogunnaike, 1994. Kosanovich and Piovoso have successfully applied linear PCA to the coefficients of Haar Wavelet transform, thus implementing a wavelet PCA analysis.

Fourier transforms and spectral signatures have been shown to offer insight into univariate data trends (Desborough and Harris, 1992) but the work has not been extended to multivariate data. Multivariate statistics literature (PCA,PLS) developed along other lines. A focus in multivariate statistics is its application to batch processing so that the outcome of a batch can be predicted or so that an abnormal batch can be detected at an early stage (for example: Nomikos and MacGregor, 1995).

In our work we make use of an efficient method, in which we create time-independent energy signatures to characterize different behaviors of the flight. The main idea is to extract a few significant combinations, i.e., principal components covering most data information of the original variables.

Chapter 3
Time-Frequency Analysis

This chapter explains, the basic objective of time-frequency analysis by explaining the Short Time Fourier Transform (STFT) and Continuous Wavelet Transform (CWT). It also discusses the advantages of using CWT over STFT. Also the key concept of *scalogram* is introduced.

3.1 Need for Time-Frequency Analysis

The need for a combined time-frequency representation stemmed from the inadequacy of either time domain or frequency domain analysis to fully describe the nature of non-stationary signals. A time frequency distribution of a signal provides information about how the spectral content of the signal evolves with time, thus providing an ideal tool to dissect, analyze and interpret non-stationary signals. This is performed by mapping a one dimensional signal in the time domain, into a two dimensional time-frequency representation of the signal. A variety of methods for obtaining the energy density of a function, simultaneously in the time and frequency have been devised, most notably the short time Fourier transform and the wavelet transform. We first take a real example to illustrate the basic idea of how time-frequency analysis can be used to clarify, and provide additional information about the behavior of a non-stationary signal before embarking on a more systematic discussion.

Using a sound signal, Figure 3.1 [12] shows three plots. Running up the page is the sound as a function of time. By examining it visually we cannot tell much, although we can clearly tell how the intensity of loudness varies with time. Below the main figure is the energy density spectrum, i.e. the magnitude squared of the

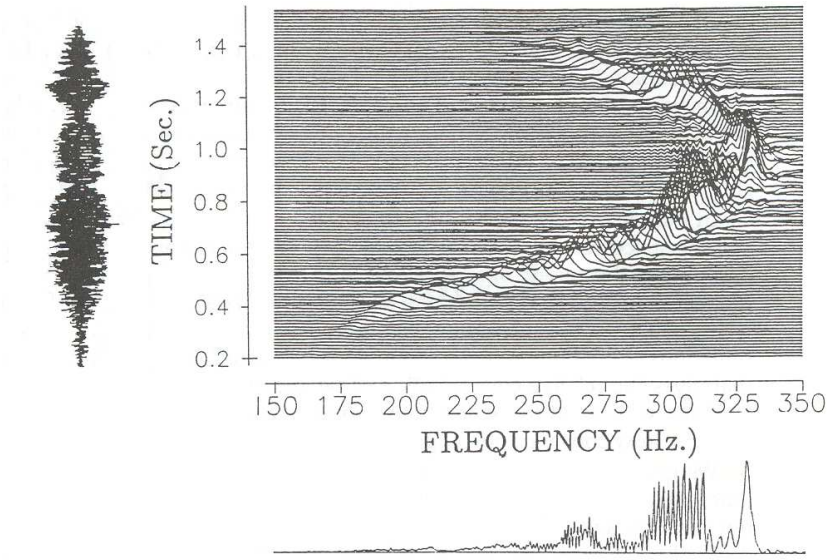


FIGURE 3.1. Time-frequency analysis of a test signal [Source: Leon Cohen, "Time-Frequency Analysis" p-72]

Fourier Transform. It indicates which frequencies existed and what their relative strengths were. For this sound, the spectrum tells us that the frequencies ranged from about 175 to about 325 cycles per second. This information is interesting and important, but does not fully describe what happened because from the spectrum we cannot know when these frequencies existed. The main figure is a time versus frequency plot, i.e. a joint time-frequency distribution. From it we can determine the frequencies and their relative intensities as time progresses. It allows to understand what is going on: at the start the frequency was about 175 Hz and increased more or less linearly to about 325 Hz in about half a second, stayed there for about a tenth of a second, and so forth.

The difference between the spectrum and a joint time-frequency representation is that the spectrum allows us to determine which frequencies existed, but a combined time-frequency analysis allows us to determine which frequencies exited at

a particular time. This is important for fault detection because, when a fault occurs their would be distortions at high frequencies and by knowing the time when these frequencies occur one could determine the time of occurrence of fault. In summary, time-frequency analysis devises a function that will describe the energy density of a signal simultaneously in time and frequency, and that can be used and manipulated in the same manner as any density.

3.2 Short Time Fourier Transform and Spectrogram

For a non-stationary signal, $x(t)$, the standard Fourier Transform is not useful for analyzing the signal. Information which is localized in time such as spikes and high frequency bursts cannot be easily detected from the Fourier Transform. Time-localization can be achieved by first windowing the signal so as to cut off only a well-localized slice of $x(t)$ and then taking its Fourier Transform. This gives rise to the Short Time Fourier Transform, (STFT) or Windowed Fourier Transform.

The STFT of a signal $x(t)$ is defined as:

$$S_t^x(\omega) = \int x(\tau)\gamma_{t,w}(\tau)d\tau = \int x(\tau)\gamma(\tau - t)e^{-j\omega\tau}d\tau \tag{3.1}$$

Note that time localization is obtained through segmenting $x(t)$ by $\gamma(\tau - t)$, the windowing function centered at $\tau = t$. The Fourier transform of this segmented signal then provides the frequency localization, which is what Fourier transform does best. The energy density spectrum at time t is defined as

$$P_{SP}^x(t,\omega) = |S_t^x(\omega)|^2 = |\int x(\tau)\gamma(\tau - t)e^{-j\omega\tau}d\tau|^2 \tag{3.2}$$

For each different time we get a different spectrum and the totality of these spectra is the time-frequency distribution, P_{SP}. This squared magnitude of the STFT is

known as the *spectrogram* and is a very common tool in signal analysis because it provides a distribution of the energy of the signal in a time-frequency plane [25].

The problem with STFT is that it provides constant resolution for all frequencies since it uses the same window for the analysis of the entire signal. If the signal to be analyzed has high frequency components for a short time span, a narrow window would be necessary for good time resolution. However, narrow windows mean wider frequency bands, resulting in poor frequency resolution. If, on the other hand, the signal also features low frequency components of longer time span, then a wider window need to be used to obtain good frequency resolution.

This was precisely the driving force behind the use of Continuous Wavelet Transform, which provides varying time and frequency resolutions by using windows of different lengths.

3.3 Wavelet Transform

One of the main ideas of wavelet transforms is to use functions different from sinusoids to approximate a function. Fourier transforms deal with just two basis functions (sine and cosine), while there are an infinite number of wavelet basis functions. The freedom of the analyzing wavelet is a major difference between the two types of analyses and is important in determining the results of the analysis. By stretching (dilating) and shifting (translating) a 'mother wavelet', one is able to capture features that are local both in time and frequency. This property alone makes wavelets more suitable for analyzing non-stationary or transient signals. In other words, wavelet basis is more interesting compared to the Fourier one because " unlike sines and cosines, individual wavelet functions are quite localized in space, simultaneously, like sines and cosines, individual wavelet functions are quite localized in frequency or more precisely characteristic scale" [35]. The wavelet transform is suited for analyzing signals that display strong transients, for example

discontinuity, or rupture. More precisely, if $x(t)$ has such singularities, these will affect only the coefficients at time points near the singularities. In contrast, the standard Fourier transform described above depends on the global properties of $x(t)$ and any singularity in it will affect all coefficients.

The continuous wavelet transform (CWT) of $x \in L_2(\mathbf{R})$ by $\psi \in L_2(\mathbf{R})$ is a projection of a function x onto a particular wavelet $\psi_{a,b}(t)$:

$$(W_\psi^x)(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t)\psi(\frac{t-b}{a})dt \tag{3.3}$$

$$x(t) = \frac{1}{C_\psi} \int_0^{\infty} \int_{-\infty}^{\infty} (W_\psi^x)(a,b)\psi(\frac{t-b}{a})\frac{da}{a^2}db. \tag{3.4}$$

where $a > 0$ and b are scale and translation parameters, respectively, ψ is the mother wavelet, C_ψ is a constant that depends on ψ, and $(W_\psi^x)(a,b)$ is the continuous wavelet transform. We can interpret 3.3 as an inner product of $x(t)$ with the scaled and translated versions of the basis functions φ:

$$(W_\psi^x)(a,b) = \int x(t)\psi_{(a,b)}(t)dt, \tag{3.5}$$

where $\psi_{(a,b)}(t) = \frac{1}{\sqrt{a}}\psi\frac{t-b}{a}$ is the dilated (by a) and translated (by b) version of the mother wavelet ψ. The most obvious difference between the Fourier transform and the CWT is that the wavelet basis functions are indexed by two parameters instead of just one. The scale a is assumed to be restricted to $\mathbf{R}^+$, which is natural since a, although tenuously, is interpreted as reciprocal of frequency. As Priestly [36] explains, this relation may be established in the case of an oscillatory mother wavelet ψ, because then as *'a'* decreases the oscillations become more intense and show high-frequency behavior. Similarly, when *'a'* increases the oscillations become drawn out and show low-frequency behavior.

In wavelet analysis, the scale that one uses in looking at data plays a special role. Wavelet algorithms process data at different scales or resolutions. If we look

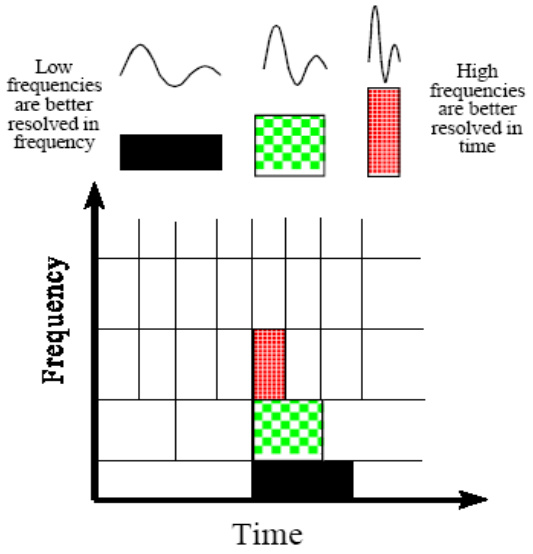


FIGURE 3.2. Wavelet Transform

at a signal with a large "window", we would notice gross features. Similarly, if we look at a signal with a small "window", we would notice small discontinuities as shown in Figure 3.2. The result in wavelet analysis is to "see the forest and the trees" [1]. A way to achieve this is to have short high-frequency fine scale functions and long low-frequency ones. This approach is known as multi-resolution analysis.

The CWT is a highly redundant transform. It is defined at all points in the time-frequency plane and the wavelet coefficients contain more information than necessary for the perfect reconstruction property to hold. By a clever discretization of the CWT one can reduce the number of wavelet coefficients to the minimum while still preserving all information of the function. By choosing a and b according to a rule that is known as the *critical sampling*

$$a = 2^j, b = 2^j k \tag{3.6}$$

one gets the discrete wavelet transform(DWT)

$$W(a,b) = W(j,k) = \int_{-\infty}^{+\infty} x(t))\psi_{j,k}(t)dt \tag{3.7}$$

where $\psi_{j,k}$ is a discrete wavelet defined as:

$$\psi_{j,k}(n) = 2^{-j/2}\psi(2^{-j}n - k)s \tag{3.8}$$

In this research, we use the continuous wavelet transform (CWT) discretized in time and scale. The discretization in time is uniform and the discretization in scale is based on the sensitivity analysis explained in Chapter 5. Continuous analysis is often easier to interpret, since its redundancy tends to reinforce the traits and makes all information more visible. This is especially true of very subtle information. The analysis using CWT gains in "readability" and in ease of interpretation.

3.4 Scalogram

The wavelet analogy of the spectrogram is the *scalogram*. Since the *CWT* behaves like an orthonormal basis decomposition, it can be shown it is isometric [18], i.e., it is an *energy preserving* transformation in the sense that

$$\int_{-\infty}^{\infty} |x(t)|^2 dt = \frac{1}{C_\psi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} |(W_\psi^x)(a,b)|^2 \frac{da}{a^2} db \tag{3.9}$$

the only constraint is that the mother wavelet $\psi \epsilon L_2(\mathbf{R})$ satisfies *admissibility condition*

$$C_\psi = \int_{-\infty}^{\infty} \frac{|\Psi(w)|^2}{w} dw < \infty \tag{3.10}$$

where Ψ is the Fourier Transform of ψ.

This leads us to define the *wavelet spectrogram*, or *scalogram*, as the squared magnitude of the *CWT*. It is a distribution of the signal in a time-scale plane associated with measure $\frac{db\ da}{a^2}$, and thus expressed in power per frequency unit, like the spectrogram[39]. However, in contrast to the spectrogram the energy of the signal is here distributed with different resolutions. The scalogram produce a more or less, easily interpretable visual two-dimensional representation of signals, where each pattern in time-frequency or time-scale plane contributes to the global energy of the signal.

Chapter 4
Overall Strategy

4.1 The Proposed Approach

In this research we use time-frequency analysis as a tool for fault detection. An occurrence of fault in a signal introduces distinctive and detectable changes in the energy distribution of the sensor data. Since the changes in energy distribution can occur at any time and keeping time information is essential, we need time-frequency analysis tools. We chose continuous wavelet transform and the scalogram to describe the energy distribution. Localization of this energy distribution describes the energy density concentrations at specific locations in time-frequency plane. This time-frequency wavelet processed data is used as indicator for further analysis. Instead of using mathematical models to describe the variations that could be expected we use empirical observation approach analyzing extensive data sets where the performance of the plane is known.

Fault detection is carried out in two phases: training phase and classification phase. In the training phase, the available sensor data are analyzed for sensitivity to abnormal behavior. This sensitivity analysis is carried out through experimental data collection of sensor reading under different conditions. Once the sensors to be used are defined, the second step in the training phase performs a second sensitivity analysis. This time the scalogram is examined as a function of the scales and only those scales that show sensitivity to various faults are retained to form the signature. The signature subspace is formed using Singular Value Decomposition (SVD) of finite dimensional matrices and selecting the most significant principal values and their associated vectors.

Since we have the prior knowledge that the incoming data should be clustered in either faulty or non-faulty classes, the fault detection problem turns into a clustering problem. In the classification phase, initially, new data coming in real time from sensor is pre-processed and an indicator matrix of the new data is obtained. Then the distance of the indicators to the signature sub-spaces is calculated and these indicators are grouped to the cluster represented by the closest signature subspace. We use nearest neighbor clustering technique for the analysis of the data obtained. A final decision is made using a majority voting mechanism taking into consideration all sensitive sensors.

With sufficient data about faulty behavior of the plane under various scenarios, we create signatures for various faults and progress towards fault isolation. We compute the distance of the indicators to each of these signatures subspaces and classify the new data to the subspace for which the distance is least.

4.2 Simulator and Data Collection

The test system used in the research and the signal processing methods applied on the sensor outputs to create indicators are discussed in this section.

The test system considered for our research purpose is a Boeing-747-100/200 aircraft system. This aircraft was chosen since its wide array of characteristics make of it the perfect representative for any of the commercial airplanes flying today and thus an ideal benchmark for FDI research. Moreover a basic model of Boeing-747 is also made available by Delft University. The Boeing-747 is an intercontinental wide body transport with four fan jet engines and designed to operate from international airports. Some of its performance characteristics are a range of 6000 nautical miles, a cruising speed greater than 965 Kmph and a design ceiling of 13716 m (see [15] for more information). Figure 4.1 shows B747-200 during flight.

FIGURE 4.1. B747-200 during flight

A comprehensive closed loop non-linear testbed model has been developed using the software package *FTLAB747*. Different flight trimming conditions (straight and level, level turn,..) can be implemented using the model. This research uses the straight and level trim and develops a methodology to detect failure in elevator, aileron, rudder, stabilizer and engine. This systematic procedure can be applied to detect any fault for any flight condition. The faults for a particular actuator can be varied by changing two parameters, namely: the intensity of fault and the time of onset of fault. Input to the system are the initial conditions, that helps to repeat the exact input for different experiments. Also duration of simulation and the integration step can be varied in the Simulink block model, that runs the simulation.

The simulations are carried out using the high fidelity model for the Boeing 747-100/200 previously trimmed at an equilibrium point. The aircraft configuration for this equilibrium point is as follows: The selected aircraft mass is 300,000 kg, and the position of the aircraft's center of gravity with respect to the (x, y, z)-axes is assumed to be 25 percent of the mean aerodynamic chord (m.a.c) for the x-axis and the point (0,0) meters for the other two axes. For normal operation of the flight, no fault is assumed, and a flight condition defined to be straight-level-flight at 7000 meters of altitude and at a true airspeed of 218 m/sec is given.

The simulink diagram of the closed loop nonlinear model is shown in Figure 4.2. The flight is simulated using the closed-loop nonlinear model for a simulation time of 60 seconds with an integration step of 0.02. For simplicity, we have not introduced any noise in the sensors or the actuators.

This model includes noise in sensors and actuators and also user defined controls to add wind and turbulence at any desired starting time. The fault generator block allows to simulate a variety of faults, such as failures in actuators (elevator, aileron, rudder, stabilizer and engine). The sensor data available from the above model are the control inputs and the state variables. The control inputs include four control surfaces plus four engines. The ten state variables of the non-linear model are: roll rate, pitch rate, yaw rate, virtual true airspeed, angle of attack, slide slip angle, roll angle, pitch angle, yaw angle, altitude. In this research we chose the state variables as the sensor measurements ignoring the measurements from the actuators. The methodology allows new data if desired.

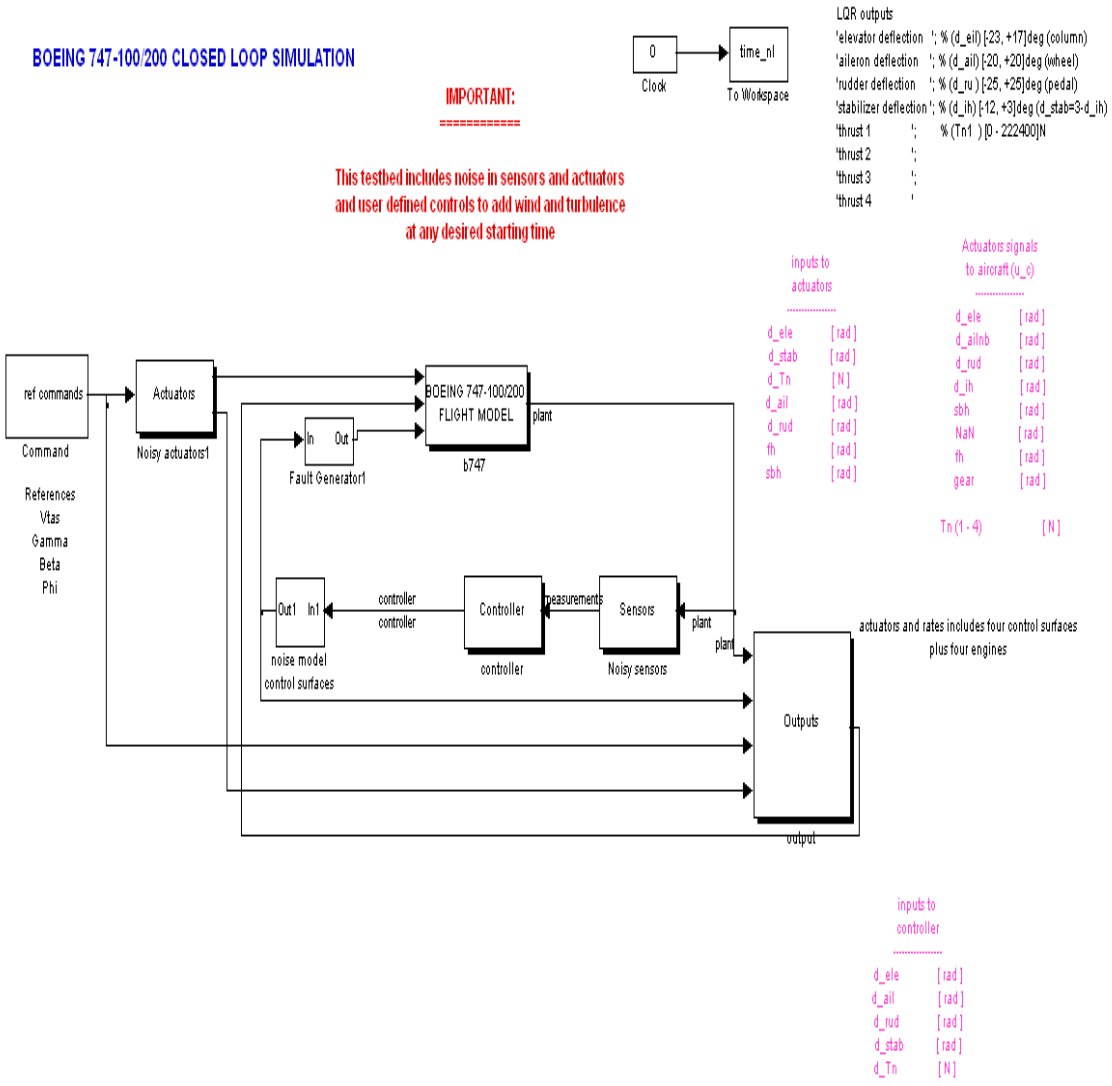


FIGURE 4.2. Closed-loop nonlinear simulink model

Chapter 5
Implementation

In this chapter we discuss in detail the implementation of the proposed approach. First, we shall discuss the two phases of fault detection. Later we shall discuss how fault detection is done for data coming in real time. As a last topic we discuss our attempt for fault isolation.

5.1 Training of Spectral Signatures

The objective of the training phase is to create signatures for normal and faulty subspaces. In the training phase we generate subspaces for each kind of fault under consideration i.e. elevator, aileron, rudder, stabilizer and engine faults. The training of spectral signatures can be divided into three stages:

- Sensitivity analysis
- Creation of Indicators
- Formation of signature sub-spaces

FIGURE 5.1. Block diagram for training phase

5.1.1 Sensitivity Analysis

Sensitivity analysis is used to reduce the dimensionality of the data and also to select scales to perform the wavelet analysis. It keeps only the important informa-

tion while discarding unwanted details. This not only speed up the fault detection algorithm, but also produce better results.

◇ **Selection of sensors**

There are 10 sensor measurements available : roll rate, pitch rate, yaw rate, virtual true airspeed, angle of attack, side slip angle, roll angle, pitch angle, yaw angle, altitude. Observations show that not all measured sensor measurements are equally sensitive to faults. We perform a preliminary sensitivity analysis of the sensors by analyzing variations in the signals from sensors under various scenarios and discarding the measurements which are of no use.To explain this let us consider an example here.

Figure 5.2 shows the 10 sensor measurements of the B747 aircraft model for an elevator failure compared to normal (fault-free) flight condition. The effect of fault is apparent in the measured signal as it can be seen in Figure 5.2. The elevator fault introduced is of 50% intensity and the fault onset time is 20 seconds.

As the flight trimming condition is straight and level trim, for the normal flight operation all the angles(roll, pitch and yaw) and their respective rates are zero. The altitude is 7000 m and the VTAS is 218 m/sec in accordance with the trimming conditions. As a fault occurs at 22 seconds all the state variables are disturbed as evident from the figure. Out of these 10 sensors the first three variables i.e. the roll rate, pitch rate and yaw rate are the derivative of their respective angles with respect to time. As is evident from the figure, the changes in the first three sensors are not distinctive and show low sensitivity to faults. They almost stay in a range near to the normal data. Hence, even though there are changes in these three sensors, we discard them for our analysis. A formal approach to examine variability of the data was not implemented.

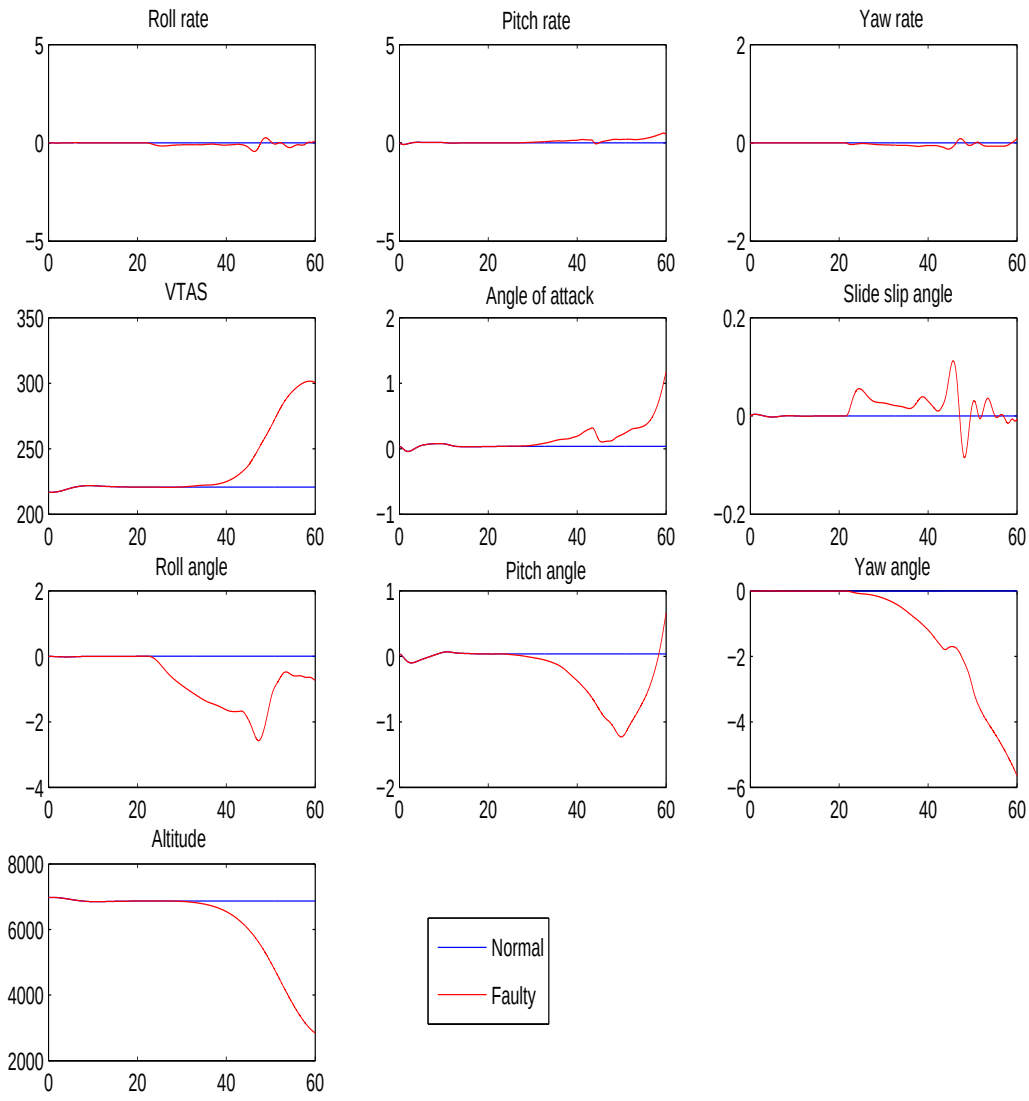


FIGURE 5.2. Sensor measurements for normal and faulty data.Fault parameters: Elevator failure, 50% intensity, time of fault: 22sec

◇ Selection of scales

The effect of fault is perfectly enhanced and represented in the pseudo power signature only if the correct scales are selected. In the case of residue-based detection, this problem is solved by using readings referred to a normal mode. We propose a new method for selection of best scales which uses all the scales that are important and sensitive to fault using the frequency domain approach. The approach captures the frequencies at which there is noticeable difference in normal and abnormal data. From these frequencies, corresponding scales for wavelet transform are computed. By using arbitrary scales, not only high frequencies but any frequency can be included if it is of importance in the data set.

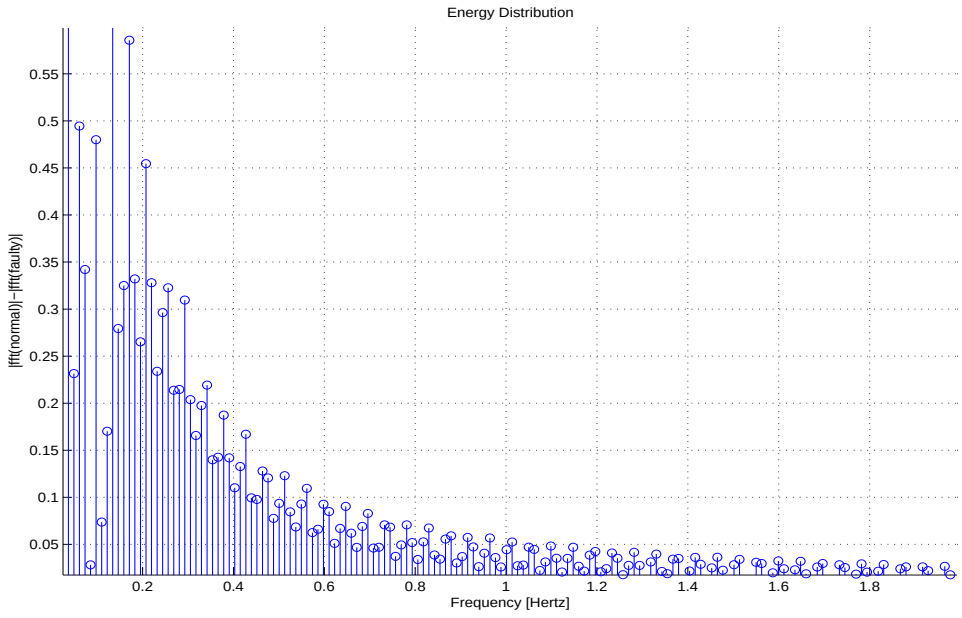


FIGURE 5.3. Plot to find the frequencies at which there is noticeable difference from normal to faulty data

Initially we create two big matrices, one for normal data and one for the faulty data which characterizes all the training data. These big matrices are obtained by concatenating all the training data into one single matrix and all the training faulty

data into a different matrix. We then analyze the energy distribution of the signal at different frequencies by computing the Fourier transform of the normal and faulty matrices obtained. Later a plot of the difference of magnitude of the Fourier transform of normal and faulty data is obtained. Using this plot the frequencies at which there is noticeable difference from normal to faulty data are measured which are the peaks in the plot. One such plot is shown in figure 5.3

Once the important frequencies are captured, the corresponding scales for the wavelet transform are obtained using the relation

$$a = \frac{F_c}{F_a} \tag{5.1}$$

where,

a is a scale

F_c is the center frequency of the wavelet

F_a is the pseudo-frequency corresponding to scale a

The best 16 scales are selected using the above procedure and used for the rest of the training phase. The matlab code for selection of scales is presented below:

```
function[a] =scales(norm,fault,state)
sd1norm=norm(:,state);
sd1norm=normc(sd1norm);
sd1fault=fault(:,state);
sd1fault=normc(sd1fault);
bN=pow2(nextpow2(length(sd1norm)));
y1 =fft(sd1norm',bN );
m1 = abs(y1);
y2=fft(sd1fault',bN);
m2=abs(y2);
m=abs(m1-m2);
fs = 50;
f=(fs/bN)*[0:bN/2-1]';
[msort, isort]=sort(m(1:bN/2));
idown=flipud(isort(:));
fbest=f(idown(1:17));
```

```
fc=centfrq('db4');
fselect=sort(fbest);
fselect=fselect(2:17);
a=fc./fselect;
```

5.1.2 Creation of Indicators

The sensor measurements obtained from the simulator are enhanced to create indicators which are used for detection of faults. These indicators make the fault more apparent to the user. Signatures are then created for normal and faulty indicators as described in the next section.

Indicators are created by applying continuous wavelet transform (CWT). As discussed in Chapter 3. CWT creates a time-frequency energy distribution. Indicators are the magnitude square of CWT known as *scalogram* (also discussed in chapter 3) given time and frequency energy distribution and are expected to change due to a fault.

The arguments to perform a CWT are: the signal to be analyzed, the scales of the analysis and the wavelet to be used. The signals to be analyzed are the sensor measurements in our case. The selection of scales for fine control over analysis is discussed in earlier section which explains the sensitivity analysis. Hereupon, in this section we shall discuss about the selection of mother wavelet for the wavelet analysis.

◇ **Selection of Wavelet**

The selection of right wavelet for analysis which enhances the changes due to a fault is very important. We should select the "mother wavelet" carefully to better approximate and capture the transient spikes of the faulty signal. The mother wavelet will not only determine how well we estimate the original signal, but also, it will affect the frequency spectrum of the faulty signal.

The Haar wavelet is the simplest wavelet in the wavelet family. It was the first wavelet introduced and is rarely used in practice because it is discontinuous and lacks energy compaction. The Daubechies wavelet is a more advanced wavelet that proved to be more appropriate for our algorithm. While the Haar wavelet does not allow for sharp transitions and fast attenuation, the Daubechies wavelet has continuous derivatives that respond well to discontinuities. Daubechies wavelets have better frequency properties than Haar wavelets. When Haar is used to decompose a signal, the wavelets cannot efficiently separate the signal into low frequency and high frequency bands. The Daubechies wavelet allows the user to decide how much fluctuation is acceptable in the high frequency bands. In most applications, the Daubechies wavelet is superior to the Haar wavelet. Empirical results obtained from our approach proved this to be true.

TABLE 5.1. Performance of different wavelets % C.D : Correct Detections %F.A :False Alarms

	Data-1		Data-2	
	%C.D	%F.A	% C.D	% F.A
Haar	87.08	12.92	89.28	10.71
Morlet	83.4	16.6	91.0	9.0
Meyer	85.24	14.76	88.48	11.52
Coif4	89.24	10.76	92.92	7.08
Bior4.4	81.04	18.95	13.32	93.32
db4	95.76	4.23	99.48	0.52

We did not confine implementing the algorithm to these two mostly used wavelets, instead, several other wavelets have been tested. The performance of the algorithm for fault detection for different wavelets is presented in table above.

The performance is tested for two different data. The data is obtained from straight and level trimming condition, at an altitude of 7000m and VTAS of 218 m/sec simulated for 60seconds. The data has no fault introduced and is free of noise. The results in table show the percentage of correct detections and false alarms. The Daubechies (dB4) wavelet outperforms all other wavelets justifying its use in our algorithm.

5.1.3 Formation of Signature-subspaces

The power content of the signal (scalogram), which forms the indicator for the signal, is computed by squaring the magnitude of the wavelet transform. The indicators so obtained are stored as a matrix. Two such matrices are formed, one containing indicators obtained from the processing of normal data and the other obtained from faulty data.

Once the scales are selected and the indicator matrices are formed, the next task is to form the signature subspaces for energy distribution which characterizes entire data. We generate power signature signal classes using SVD of finite dimensional matrices. The methodology is based on a Principal Component Analysis (PCA) [21] technique, and is derived from the decomposition of the CWT of a signal as a sum of separable terms.

The entry (i, j) of the indicator matrix is the value of the scalogram for scale a_j computed at time t_i. The outline of the procedure to form the time-independent signature for the scalogram is the following: Let $\boldsymbol{M}$ be the matrix of scalograms where each row corresponds to one of the scale selected and each column is associated to a time instant. Using the singular value decomposition one can always write

$$M = \sum_k \sigma_k u_k v_k^T \tag{5.2}$$

where σ_k are the singular values for the matrix $\boldsymbol{M}$ and u_k, v_k are unitary vectors - the collection of the $u_k(/v_k)$ forms a unitary matrix. The PCA selects a small number of "significant" singular values to approximate the matrix data.

Each row of the scalogram represents the energy distribution for a given window and is used as the distribution at the time corresponding to the center of the window.

Every row of $\boldsymbol{M}$ is of the form

$$row_k = \sum_{i=1}^{r} u_{ik}\sigma_i v_i^* \tag{5.3}$$

which is equivalent to

$$row_k \in span\{v_1^*, v_2^*, ...v_r^*\} \tag{5.4}$$

The 'r' orthogonal vectors of the matrix V form the orthonormal basis of a 'r' dimensional space. Hence the subspace is defined as the signature subspaces for all rows, for our case the signature sub-space for energy distribution. This above approach is applied to get two subspaces, normal and faulty formed from normal and faulty indicator matrices respectively. These subspaces are defined as the *signature subspaces* for the collection of signals.

Figure 5.4 shows that there is more than one significant value for the indicator matrix. The first value is high but the following are not negligible. The number 'r' of significant singular values/vectors can be judged by examination of the magnitude of the singular values and the magnitude of the autocorrelation of each row of V^T [28]. We considered 8 singular values (and the associated basis vectors) to form the signature subspace; this number is chosen after an exhaustive search because they yielded the best results considering different faults.

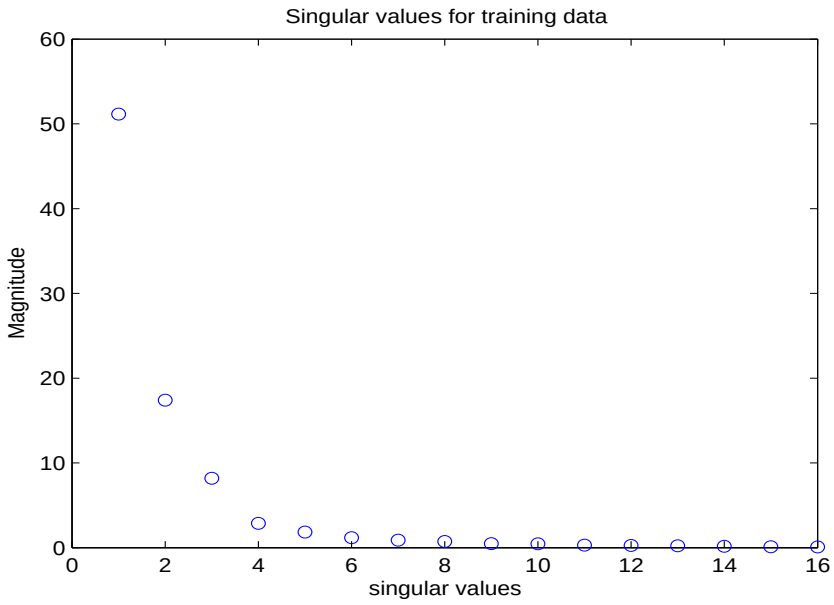


FIGURE 5.4. Singular values for training data

5.2 Classification of Unknown Signatures

The objective of the classification phase is to use the trained spectral signatures for classifying a new data obtained from the sensor into either normal or abnormal class. Figure 5.5 shows the block diagram which represents the classification phase of fault detection. The classification phase consists of:

- Removal of Edge effect
- Creation of Indicators
- Computation of distance clusters
- Classification of Indicators
- Decision making using Voting mechanism

5.2.1 Removal of Edge Effect

Abrupt boundary truncation of data introduces artifacts in the processed data and often cause distortion of patterns in the frequency domain. These artifacts

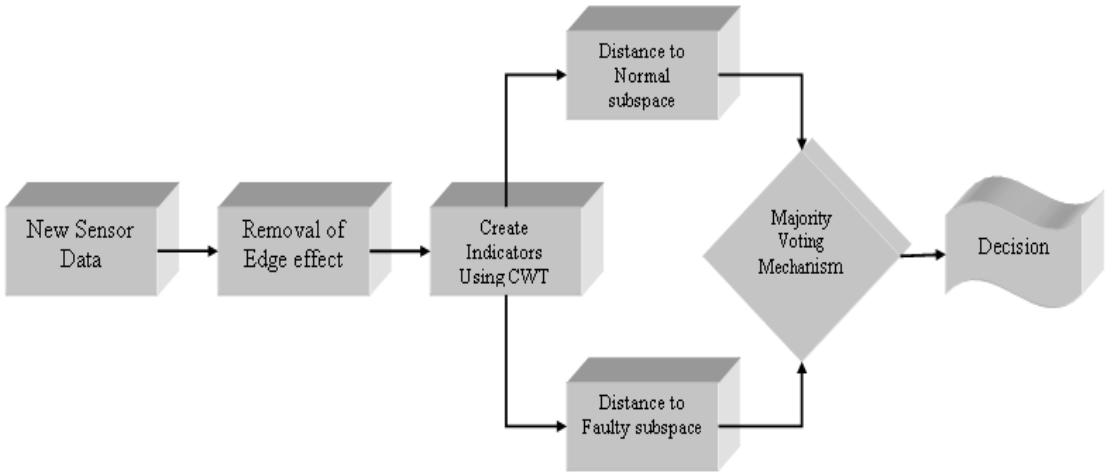


FIGURE 5.5. Block diagram for classification phase

that occurs at the first and the last few samples of a sequence are called edge-effects. These artifacts due to edge effects may adversely affect the result of data analysis: the effects can be significant depending on edge abruptness. These effects are often neglected in one-dimensional time signal due to the length of the signal[2]. In signal processing, where the "length" of the signal is much smaller, these effects are an important issue and hence, signal edges should be handled with care.

Traditional solution to reduce edge effects is to smooth the boundary of the signal prior to applying the transformation using special window functions such as Hamming or trapezoidal windows. This is followed by zero-padding, that smoothes the boundary by adding zeros on the outside. This method improves the results but still distorts the signal. Another method [3] to deal with edge effect is by reflecting the original signal about its boundaries thus extending the signal. This procedure is simple and exploits the natural property of "circular" or periodic convolution.

Both the above methods have been considered and not much difference is noticed among the methods. Their performance is almost equal in our case and the results

show that the false alarm/missed alarm rate decreased by 3% compared to results without using any edge effect removal method.

5.2.2 Creation of Indicators

The basic idea is to develop a signature that characterizes the energy distribution of a signal in manner that is essentially independent of the duration of the signal. We create this signature by defining an "instantaneous energy distribution". Indicators are created as described in section 5.1.2.

5.2.3 Clustering Using Nearest Neighbor rule

The nearest neighbor (NN) rule [16] classifies an unknown sample into the class of its nearest neighbor, according to some similarity measure (a distance). Given a distance, it is very simple to build up a classifier based on this rule, which often obtains very good classification rates, many times better than those of other more complex classifiers. The easiest implementation of the NN rule consists on computing all the distances between the sample and each object (prototype) in the training set, and then the nearest neighbor is the prototype whose distance is the minimum.

Figure 5.6 shows two 2-Dimensional subspaces, one for normal and one for abnormal. To classify the incoming new indicator as belonging to normal or faulty class, the distance of the indicators from both the subspaces is compared and the closest vector subspace signature identifies the condition of the indicator.

Computation of distance clusters

Given orthonormal basis $\{V_1, V_2, V_3\}$ the projection $\hat{p}$ of the test data onto the subspace is computed as

$$\hat{p} = \alpha_1 V_1 + \alpha_2 V_2 + \alpha_3 V_3 \tag{5.5}$$

where,

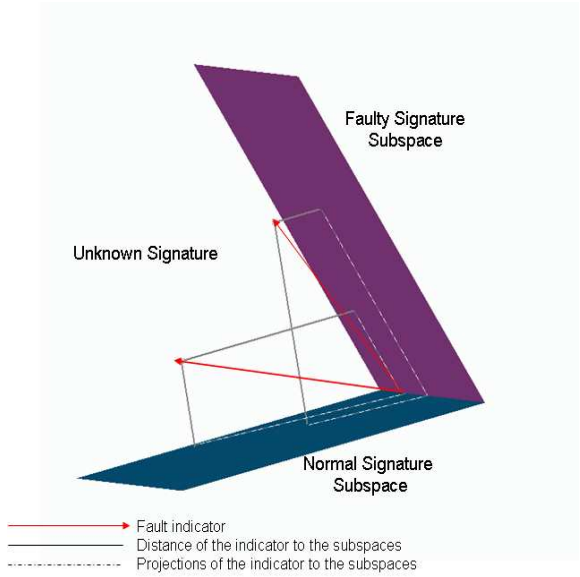


FIGURE 5.6. Concept of subspaces

$$\alpha_1 = \langle p_1, V_1 \rangle$$

$$\alpha_2 = \langle p_2, V_2 \rangle$$

$$\alpha_3 = \langle p_3, V_3 \rangle$$

Once the projection is computed, the difference between the indicator vector and its projection on the subspace gives the distance of the indicator to the subspace.

Classification of Indicators

Depending on the distances the data is declared normal or faulty according to the following decision making criterion

if

$$dist_1 > dist_2 \text{ ; data is faulty}$$

$$dist_1 < dist_2 \text{ ; data is normal}$$

where,

$dist_1$ is the distance to normal sub-space

$dist_2$ is the distance to faulty sub-space

Decision making based on majority voting mechanism

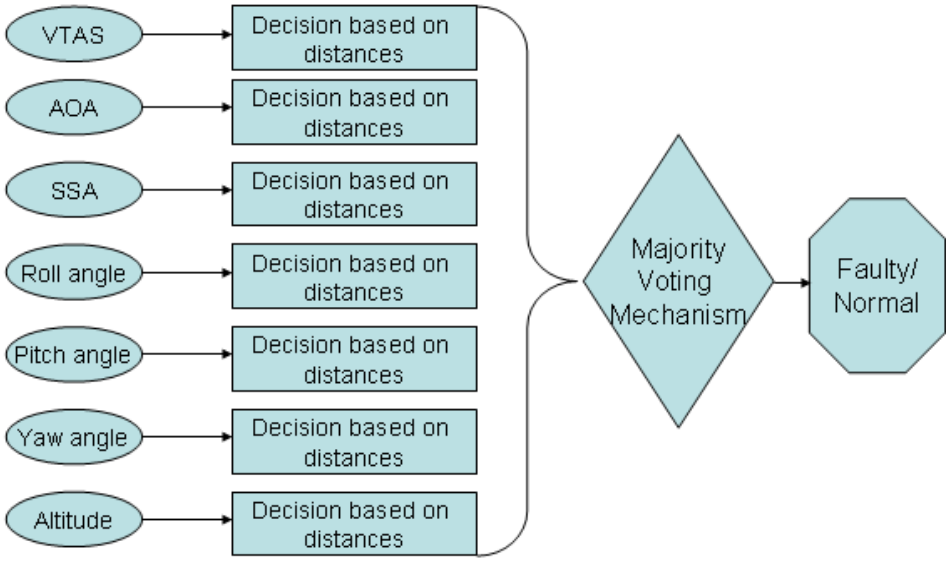


FIGURE 5.7. Decision making

It is observed that not all the states obtained from the B747 model are indicative of fault. Hence only the states which are more indicative to changes are considered and then a voting mechanism is put, in order to decide whether the test data from sensors is normal or faulty at a particular instant of time. If a majority of states indicate faulty we declare the data as faulty and if majority of states declare normal, then the data is declared as normal. The states considered are '*True airspeed*', '*Angle of attack*', '*Slideslip angle*', '*Roll angle*', '*Pitch angle*', '*Yaw angle*', '*Altitude*'. Figure 5.7 shows the decision making algorithm considering these 7 sensors and employing a majority voting mechanism.

The final performance evaluation was then computed as the percent of correctly identified time instances over the entire signal length. The three categories into which the performance is classified are:

- Correct detections:

$$\%Correct\ detections = \frac{\#of\ correctly\ detected\ time\ instances}{total\ \#of\ time\ instances} \times 100 \quad (5.6)$$

- False alarms:

$$\%\ False\ alarms = \frac{\#of\ time\ instances\ normal\ data\ is\ detected\ as\ faulty}{total\ \#of\ time\ instances} \times 100 \quad (5.7)$$

- Missed alarms:

$$\%\ Missed\ alarms = \frac{\#of\ time\ instances\ faulty\ data\ is\ detected\ as\ normal}{total\ \#of\ time\ instances} \times 100 \quad (5.8)$$

5.3 Online Fault Detection

Until now we have discussed the fault detection strategy by considering the entire length of signal. However, this is not always practical in real life. The reason being that, suppose the time of simulation is 100 seconds and a fault occurs at 50 seconds. It is not practical to wait until the 100^{th} second to declare a fault. But instead we need to detect the fault as soon as it occurred and necessary control mechanism should be operated to make sure that no disaster take place later. It is very important to have at our disposal, automatic methods for detecting faults in real-time. Implementation of online fault detection by using windowing on the experimental data is shown below.

The Continuous Wavelet Transform operates on finite length (length N) sequences. In this research, the CWT is applied to a variety of window lengths and the process of fault detection for that particular window is carried out. The process is continued on the next consecutive window of N points and the CWT followed by the fault detection procedure is carried out. There is an issue however in applying

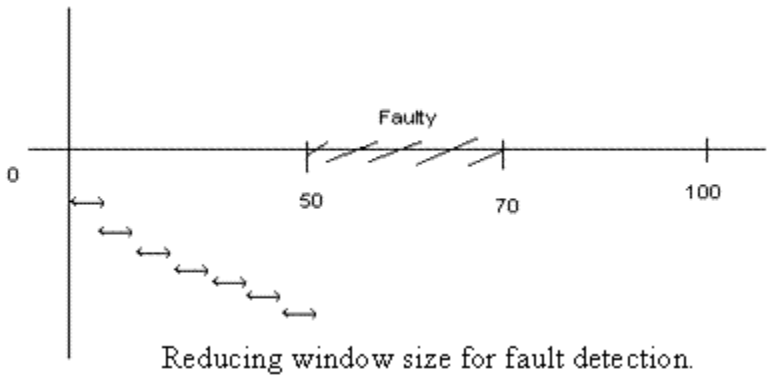


FIGURE 5.8. Overlapping windows for online fault detection

CWT. If the shape of window is rectangle it creates abrupt discontinuities at the ends. For this research, no window function was used (implying a rectangular window function), as the incoming data itself is constantly changing. To reduce the edge effects because of rectangular window, we adopt the mirroring method and get rid of the edge effects. To further reduce the waiting time for fault detection, overlapping windows are considered as shown in Figure 5.8.

The size of the window is determined according to the condition of the signal i.e. different faults have different window sizes. The graphical user interface of the algorithm allows the user to define the window size and also the window slide length. However, a default window size along with window slide-length are specified which gives best results. The real-time fault detection is approached in the following way.

In the training phase, the spectral signature subspaces for different conditions are obtained by performing singular value decomposition on a big matrix 'S'. This big matrix 'S' is obtained by concatenating the indicator matrices of each window of the entire signal. Once the signatures are created, then each window is declared normal/faulty according to the nearest neighbor rule clustering. The procedure is

similar to the one discussed in the above sections. If any of the windows declare a fault, an indication will be sent immediately to the pilot, who would take the necessary action and control the flight. The leading edge of a particular window should be considered as the time of fault detection because using this we declare a fault based on the previous data of the signal. Whereas, if we consider the trailing edge as the fault detection time, we base our decision depending on future data which is not feasible.

5.4 Fault Isolation

If the system is normal, i.e. there are no faults in the system, then the sensor measurements will be within an acceptable region near the trimming conditions. A fault in the system can be detected by comparing the new sensor data with predefined normal and faulty signatures. If the new data is closer to the faulty subspace, the system is declared faulty. Once the system is declared faulty, the fault isolation should be followed, i.e. we need to identify the fault location.

As discussed earlier, we considered five faults: elevator, aileron, rudder, stabilizer and engine failures. The task is to categorize a new sensor data into one of these faults. Here, fault isolation can be considered as an extension to fault detection discussed in earlier sections. Besides normal flight operation also training data from all possible faulty operations are collected. For each of the fault, a signature subspace is created. Also a signature subspace is formed for normal data. Once the six subspaces are created, the fault isolation is followed. For the purpose of fault isolation we compute the distance of a new sensor data (indicators) to each of the above six signature subspaces. Likewise, for a particular window, we also compute distance to six subspaces for each of the most sensitive sensors. A final decision is made if majority of the sensors declare it belonging to a particular fault. The formation of spectral signatures and computation of distances is similar to that

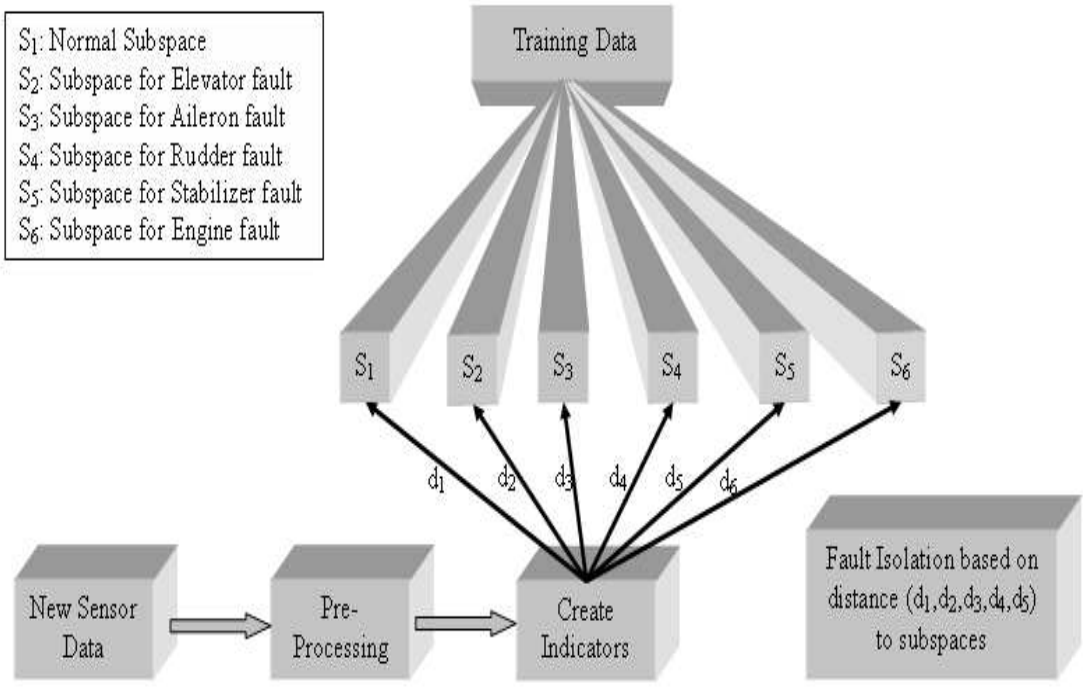


FIGURE 5.9. Fault Isolation concept

discussed earlier and hence is not repeated here. A symbolic fault isolation scheme is shown in Figure 5.9.

Chapter 6
Verification of Proposed Methodology

This section presents the experimental verification of the proposed fault detection method for data obtained from the B747 aircraft model. The experimental study has three parts. In the first set, we considered entire data as a single window. The second set of experiments verifies the method for online fault detection. The third set of experiments corresponds to fault isolation.

6.1 Results Considering Single Window

In this section we present the results obtained by considering the entire signal as a single window. We compute the percentage of correct detections using the Equation 5.6. All the data considered here have only one particular fault and the fault onset time is 0 seconds. The faults considered are faults in actuators : elevator, aileron, rudder, stabilizer and engine. In order to compare the results obtained all the data considered have same initial conditions. Also an analysis on normal data (fault-free) has been performed to check the effectiveness of the algorithm for false alarms. The normal data detection was good with 85% correct detections. The plot for the normal data is shown in Figure 6.1 which gives the percentage of correct detections for different test data. As an engine failure is reflected in all the sensors, the detection was almost complete. The following graphs show the percentage of correct detection as a function of percentage loss of efficiency for different faults.

6.2 Online Fault Detection

We report here the results of a typical set of experiments for online fault detection. For each actuator we introduce a loss of efficiency fault at different random times and of different magnitudes to get a set of data for each actuator. For the simu-

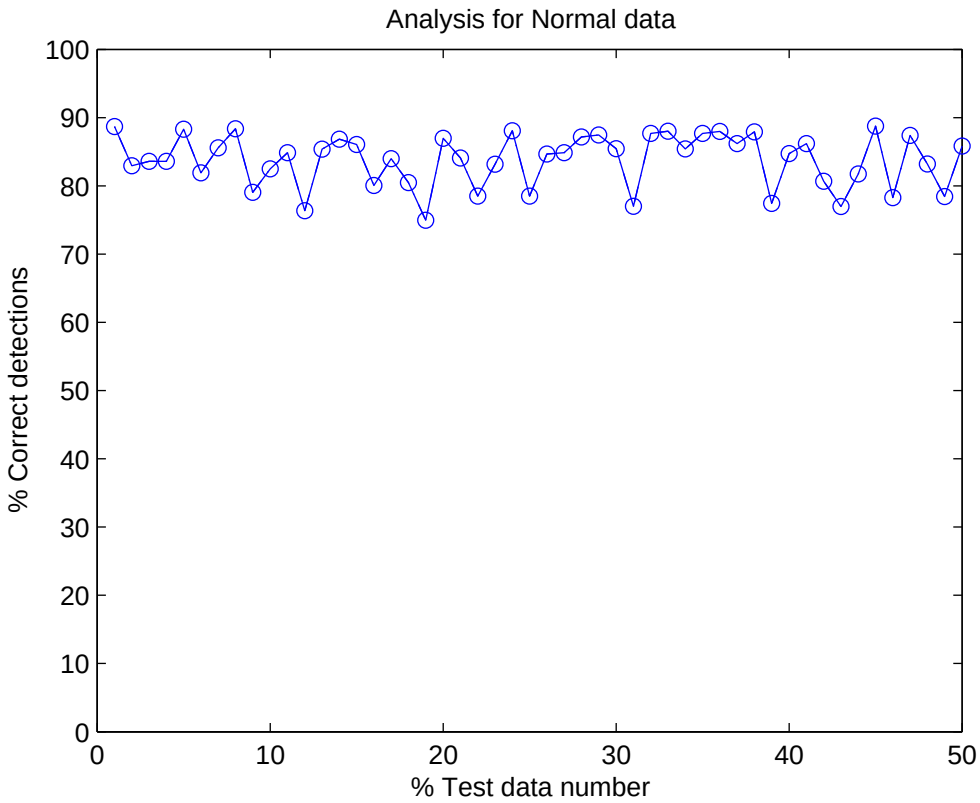


FIGURE 6.1. % correct detections for different normal data

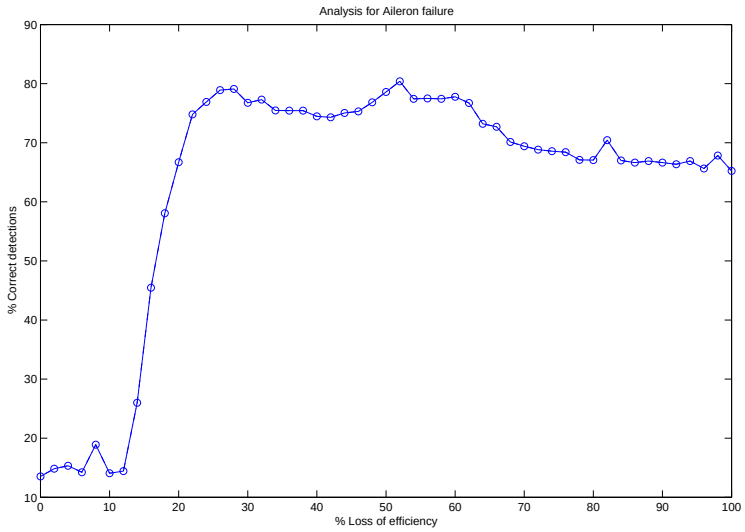


FIGURE 6.2. % correct detections as a function of %loss of efficiency. Fault parameters: aileron failure, time of fault: 0sec, loss of efficiency: [0:2:100], single initial state

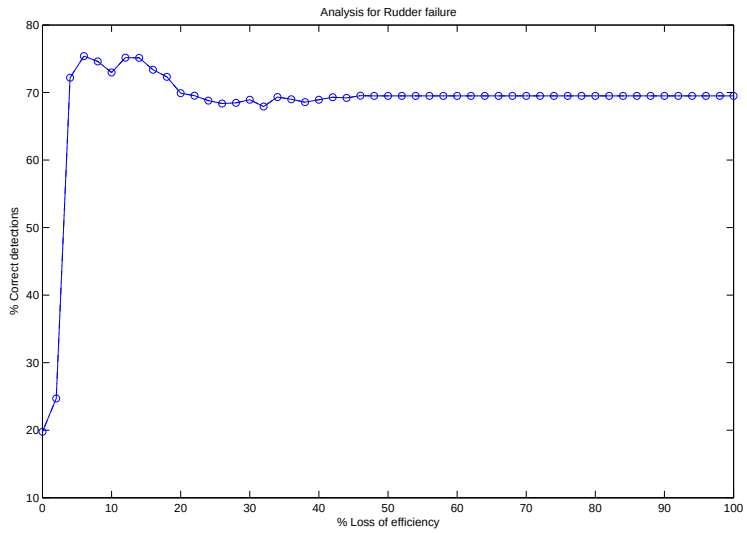


FIGURE 6.3. % correct detections as a function of %loss of efficiency. Fault parameters: rudder failure, time of fault: 0sec, loss of efficiency: [0:2:100], single initial state

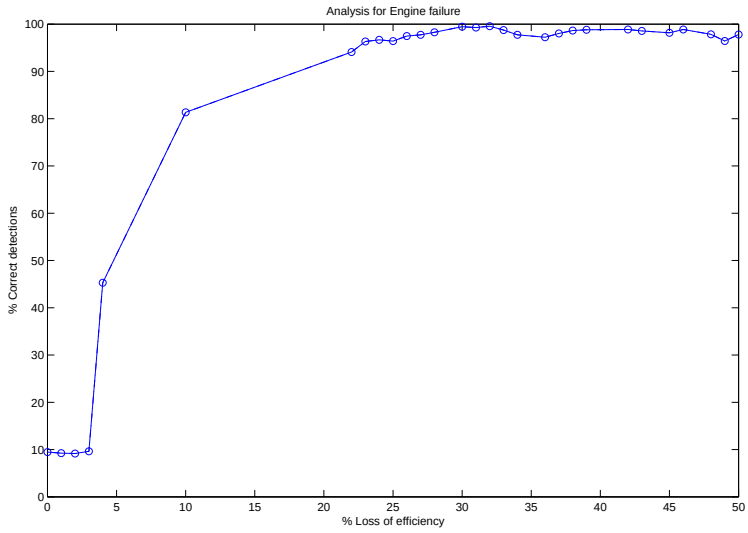


FIGURE 6.4. % correct detections as a function of %loss of efficiency. Fault parameters: engine failure, time of fault: 0sec, loss of efficiency: [0:2:50], single initial state

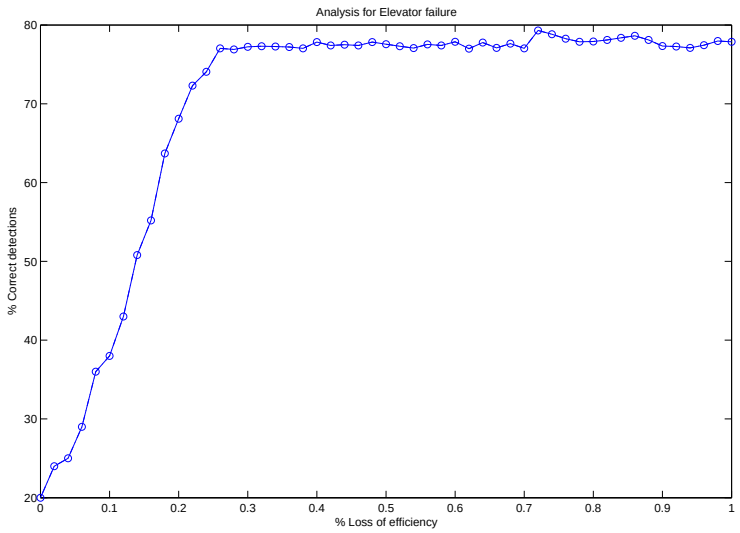


FIGURE 6.5. % correct detections as a function of %loss of efficiency. Fault parameters: elevator failure, time of fault: 0sec, loss of efficiency: [0:2:100], single initial state

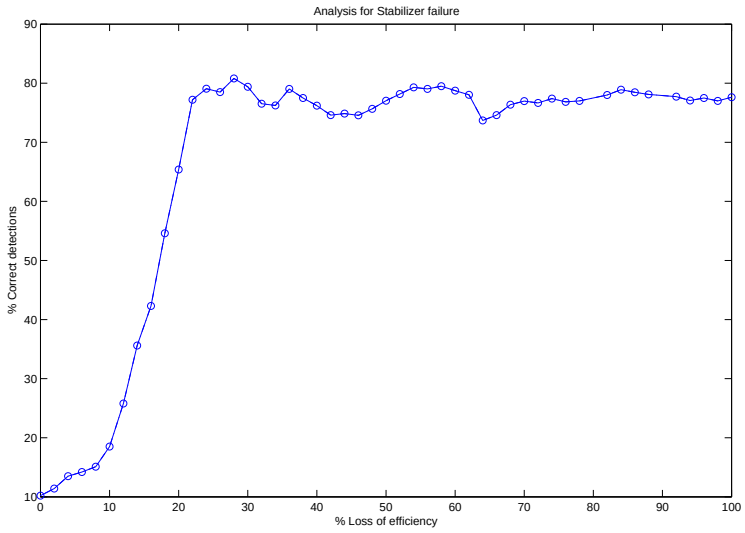


FIGURE 6.6. % correct detections as a function of %loss of efficiency. Fault parameters: stabilizer failure, time of fault: 0sec, loss of efficiency: [0:2:100], single initial state

lations performed for these tests, the plane was disturbed by adding a change to the initial state and then allowing the control system to return it to steady state. All the experiments have the same initial disturbances for comparison. The most difficult fault to detect was in the stabilizer where the mean time to detection was 9.58 sec with a standard deviation of 5.25 sec. The table below gives the statistics for all four control surfaces when the online fault detection method is used.

TABLE 6.1. Performance of model-free fault detection for various faults

	Mean Detection delay(sec)	Standard Deviation (sec)
Elevator	9.4356	4.4613
Aileron	4.5145	4.8414
Rudder	8.2136	5.2644
Stabilizer	9.5897	5.2482

The graphs of detection time distribution (Figure 6.7) show very interesting properties. There is a large number of cases where the fault is not detected i.e. missed alarm. A study of these results shows a strong correlation with the time of the fault and also percentage loss of efficiency. Specifically, if the plane is in perfect trim and there is a loss of efficiency in one actuator, the fault may not be detected until the actuator is required to operate; i.e., until the plane is in a transient state. There were cases when the fault was introduced very late and the plane was essentially back in steady state. Those cases seem to account for the missed alarms. If the loss of efficiency of the fault is not significant it accounts to missed alarms. A more clear understanding of this correlation with time of fault and loss of efficiency is given next using the below plots.

Figure 6.8 shows the scatter plot for fault detection of an elevator failure. It is a three dimensional view all the data displaying the detection delay varied with

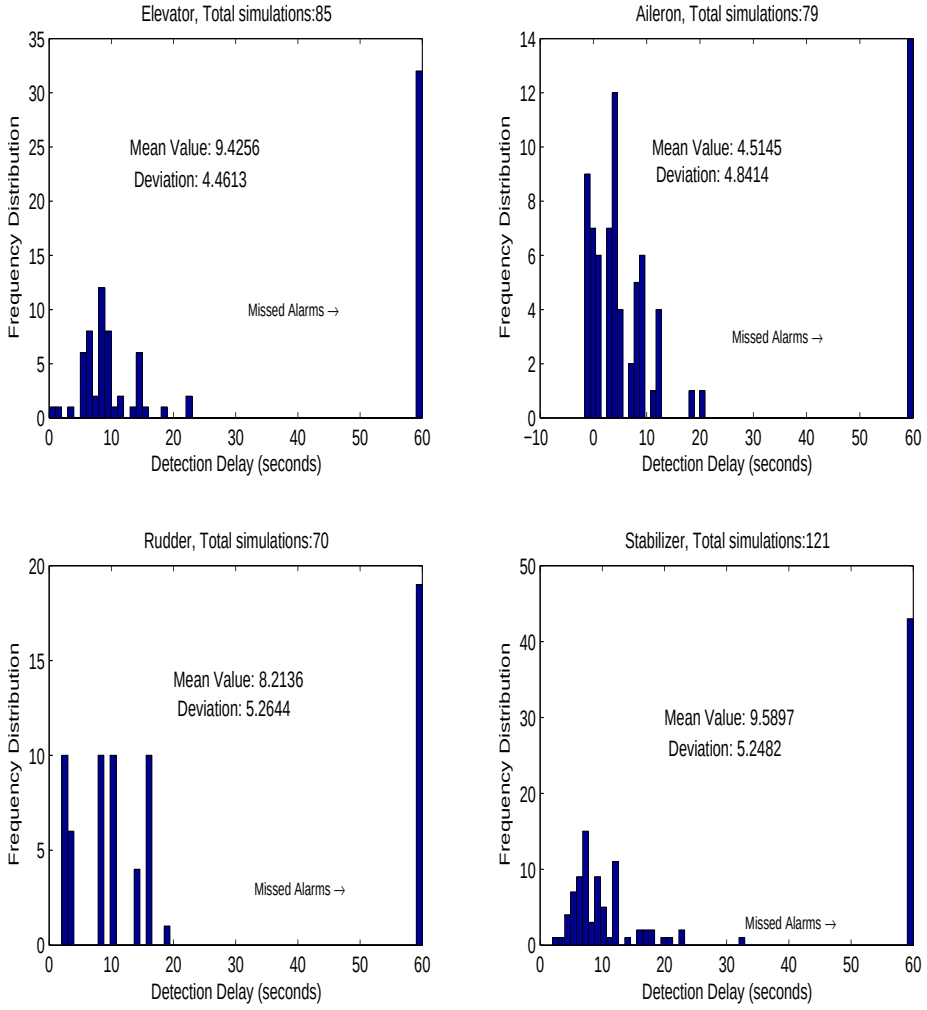


FIGURE 6.7. Distributions of fault detection time using model free fault detection method for different faults: elevator failure, aileron failure, rudder failure and stabilizer failure.

respect to loss of efficiency and fault onset time. The same figure is shown in a more clear way in two different figures 6.9 and 6.10. Figure 6.9 shows how the detection delay varies with loss of efficiency, grouping data with same fault onset time into one group (shown in legend). Each fault onset time is shown with different color so that we can analyze for which data the detection is good. We say the fault detection is good if the detection delay if small. As it is clear from the graph the detection delay decreases as the intensity of fault increases. The yellow points show that the algorithm could not detect a fault which occurred at a time of 40 seconds. This is reasonable because the fault was introduced very late and the plane is back to steady state and if a fault occurs much later, it is more difficult to detect a fault.

Figure 6.10 shows how the detection delay varies with fault onset time grouping data with same loss of efficiency into same group (shown in legend). Each % loss of efficiency is represented by a unique color. The data that could not be detected

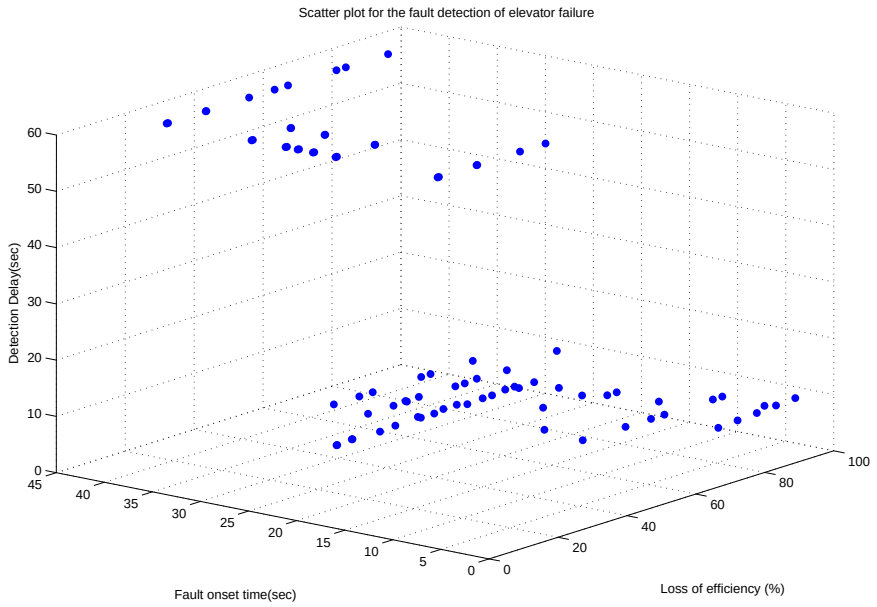


FIGURE 6.8. Scatter plot for fault detection of an elevator failure. Detection delay as a function of both fault onset time and intensity of fault. Length of window:12 sec Window slide: 4sec Total simulations:85

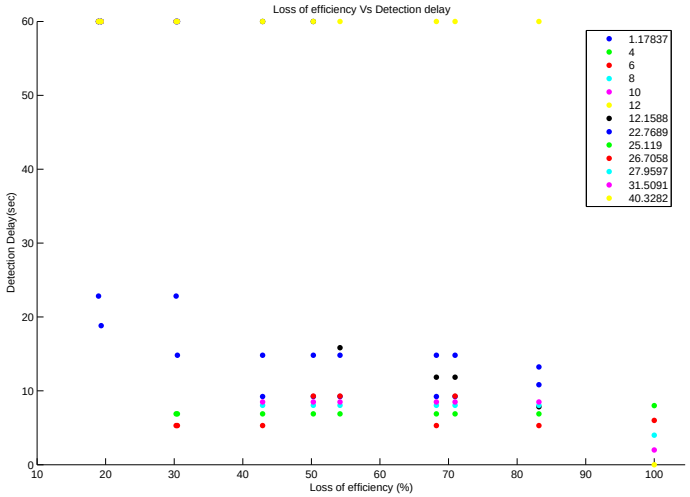


FIGURE 6.9. Change in detection delay as a function of loss of efficiency grouping same fault onset times

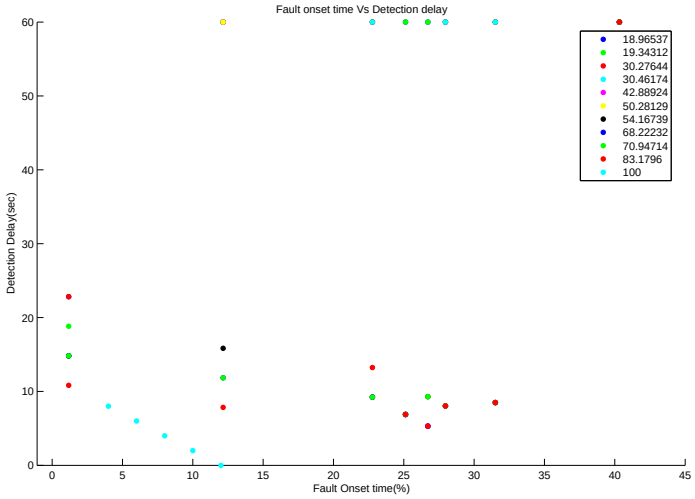


FIGURE 6.10. Change in detection delay as a function of fault onset time grouping same loss of efficiency

in this case is the data with fault intensity smaller than 35%. This is reasonable because, as the fault is less predominant, there would not be any noticeable change in the energy distribution and hence cannot be detected easily and vice versa for fault with high intensity.

The following plots are representative of the type of results that are obtained for different faults using the described fault detection method.

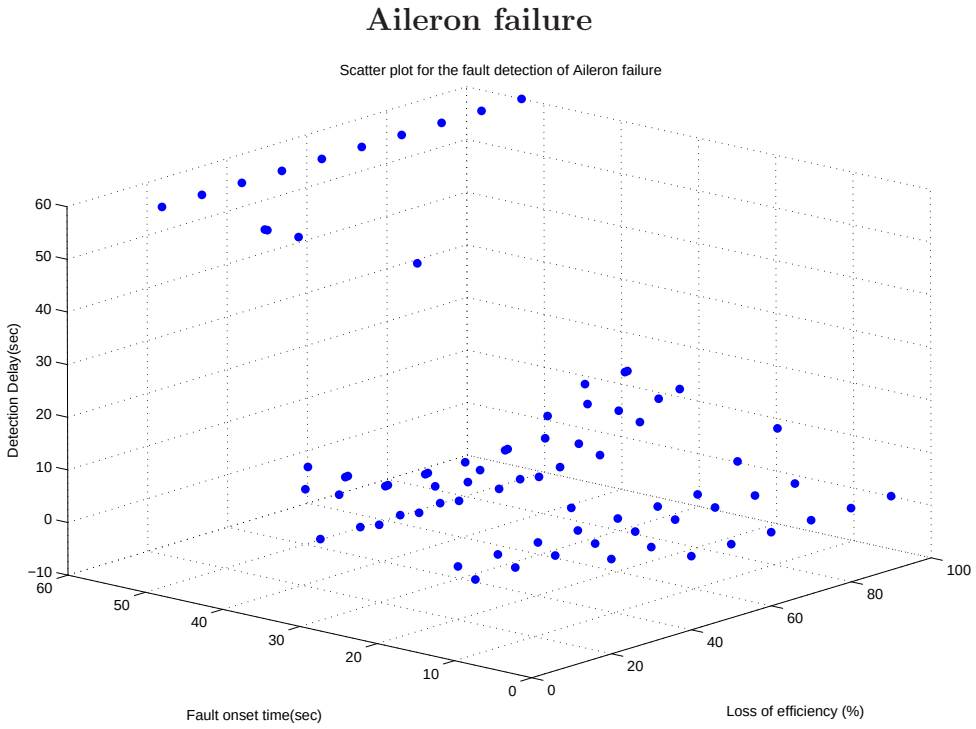


FIGURE 6.11. Scatter plot for fault detection of an aileron failure. Detection delay as a function of both fault onset time and intensity of fault. Length of window:12 sec Window slide: 4sec Total simulations:79

6.3 Fault Isolation

For all the data used above we also tested the fault isolation strategy described in section 5.4. The results show that the fault isolation algorithm is not efficient. There are many missed alarms and also most of the data was mis-classified. Most of the data was wrongly classified as rudder failure. The results of 355 simulations

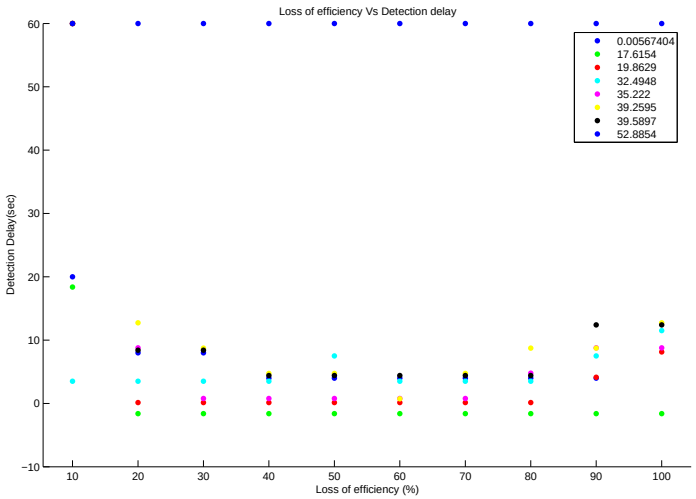


FIGURE 6.12. Change in detection delay as a function of loss of efficiency grouping same fault onset times

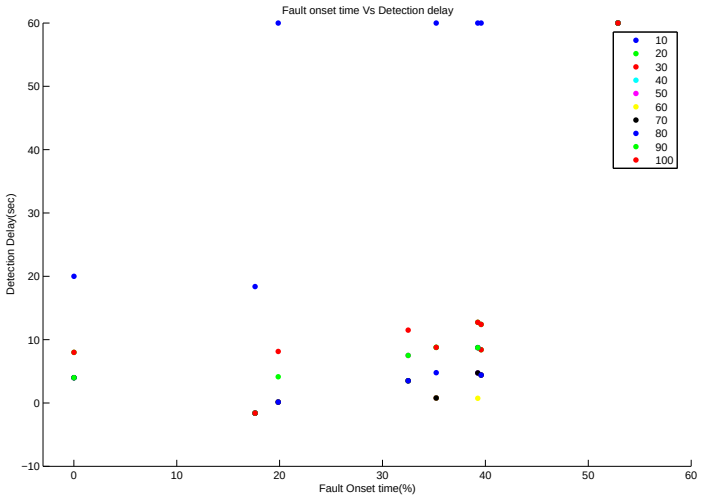


FIGURE 6.13. Change in detection delay as a function of fault onset time grouping same loss of efficiency

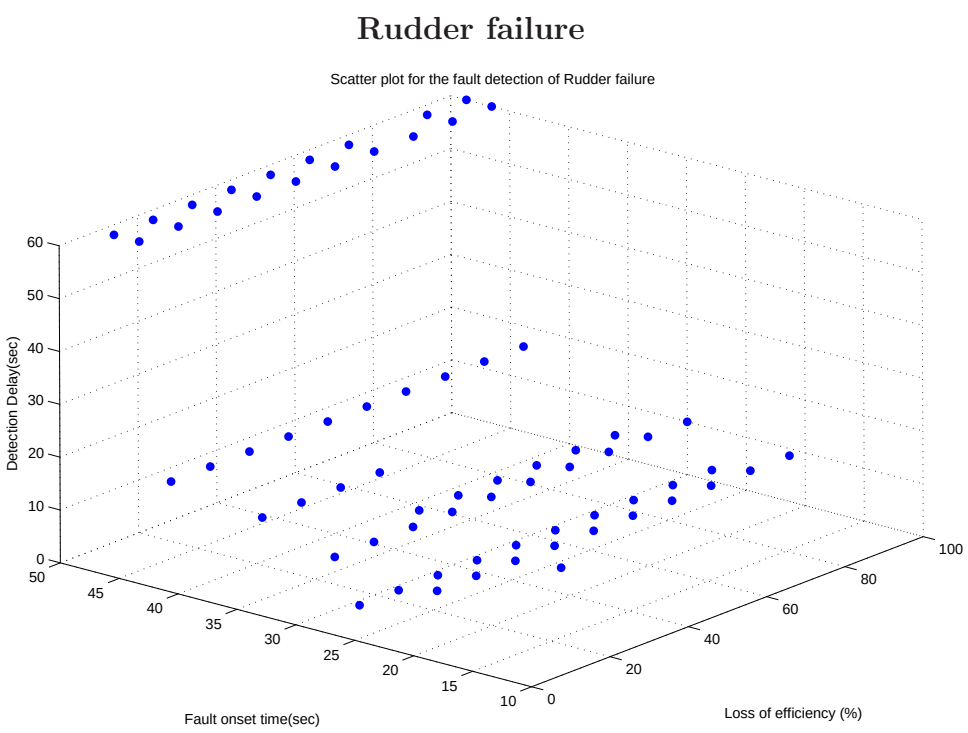


FIGURE 6.14. Scatter plot for fault detection of rudder failure. Detection delay as a function of both fault onset time and intensity of fault. Length of window:20 sec Window slide: 10sec Total simulations:70

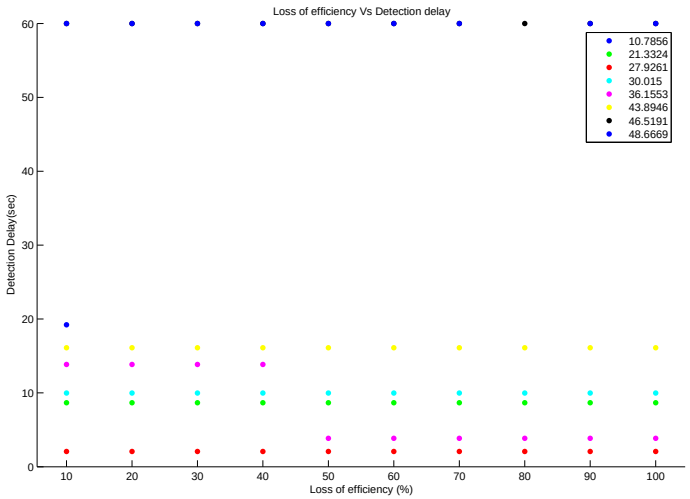


FIGURE 6.15. Change in detection delay as a function of loss of efficiency grouping same fault onset times

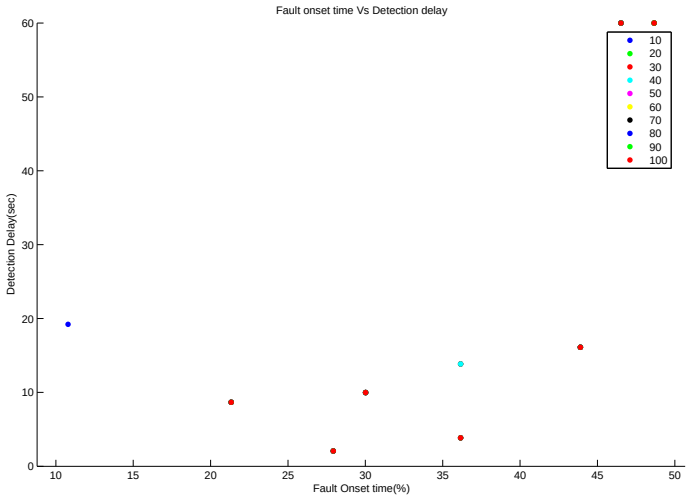


FIGURE 6.16. Change in detection delay as a function of fault onset time grouping same loss of efficiency

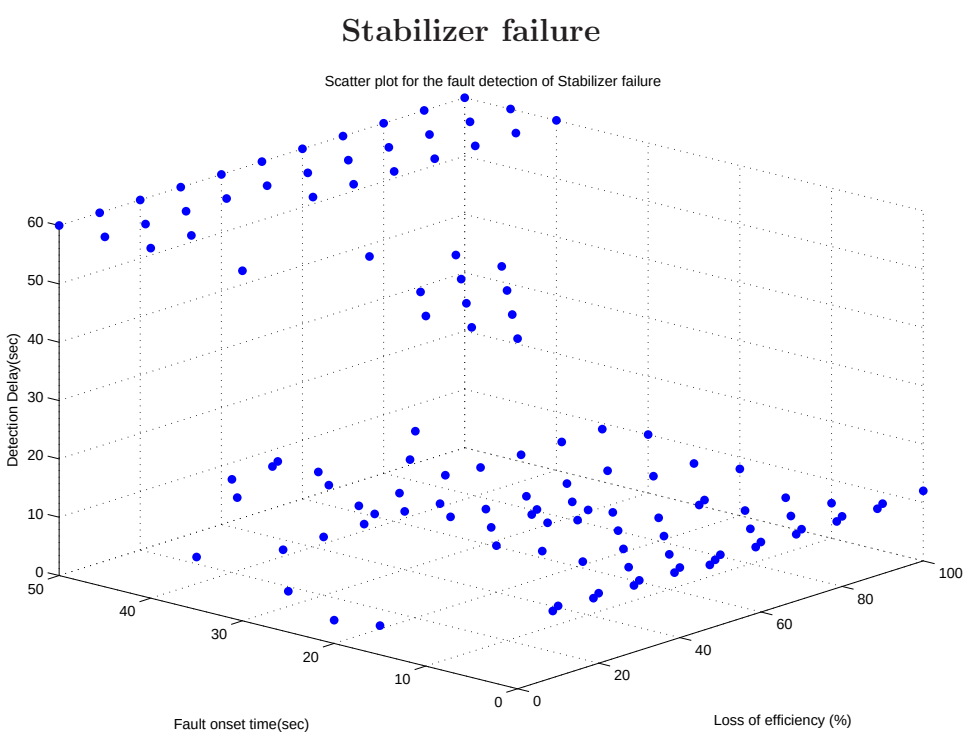


FIGURE 6.17. Scatter plot for fault detection of a stabilizer failure. Detection delay as a function of both fault onset time and intensity of fault. Length of window:12 sec Window slide: 4sec Total simulations:121

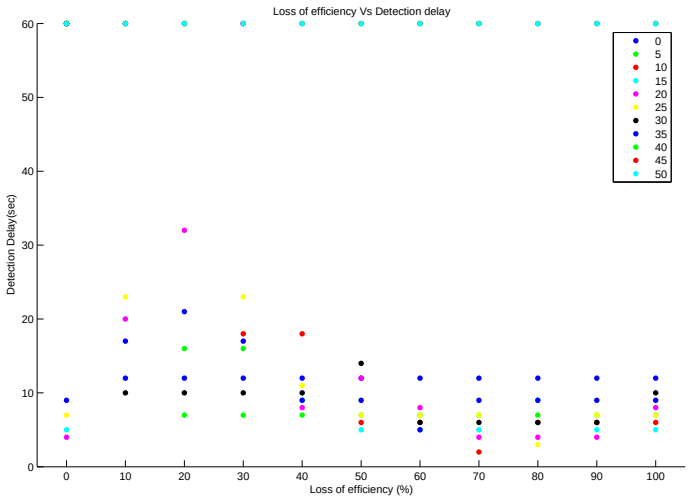


FIGURE 6.18. Change in detection delay as a function of loss of efficiency grouping same fault onset times

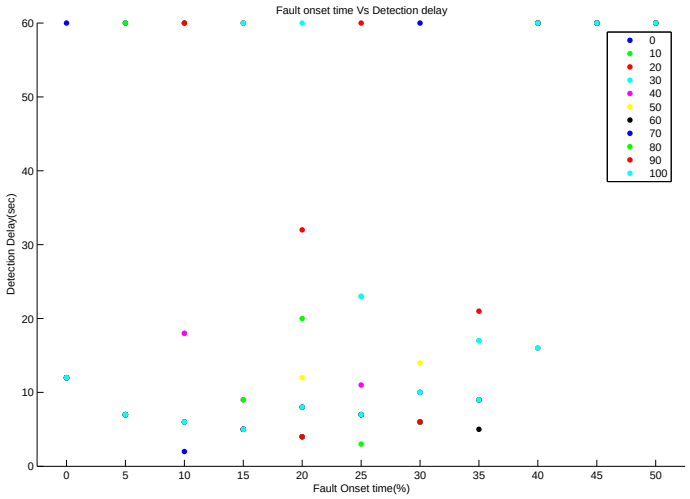


FIGURE 6.19. Change in detection delay as a function of fault onset time grouping same loss of efficiencies

are summarized in the next figure. Each actuator is identified by a unique color so that we can show not only the number of wrong detections but also what was detected. Thus for example, in the case of faults in the elevator we have correctly identified the event in 33% of the cases, missed the fault in 25% of the cases and 42% was wrongly identified as a rudder fault.

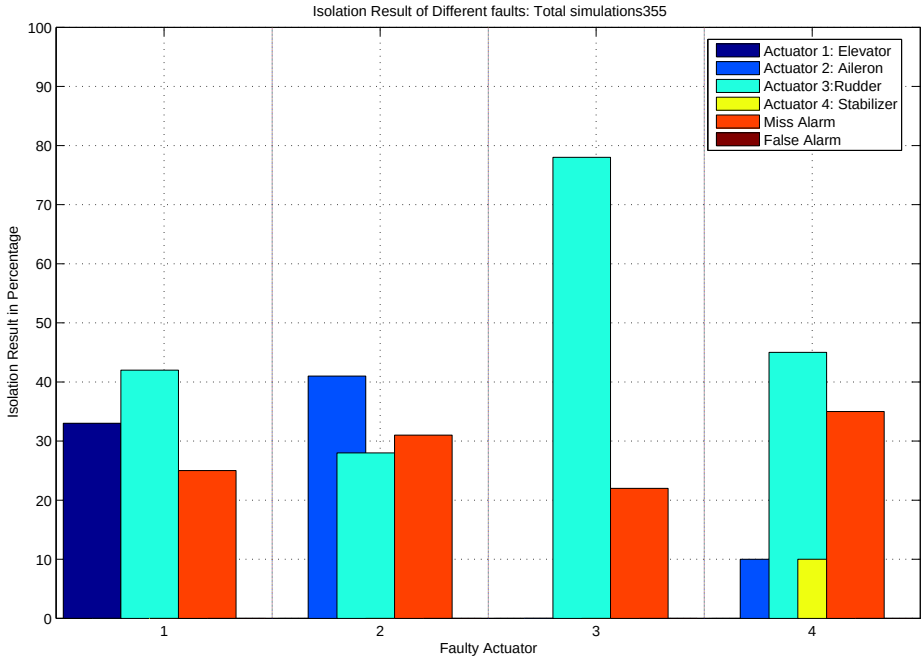


FIGURE 6.20. Fault isolation experiments. Results of 355 simulations(85 for elevator, 79 for aileron, 70 for rudder and 121 for stabilizer. Multiple bars identify actuator was wrongly detected as faulty

Chapter 7
Conclusions

This research studies the "model free" fault detection using time-frequency analysis. This research developed signatures based on spectral energy density for two conditions, normal and faulty condition where, the faults considered are the faults in elevator, aileron, rudder, stabilizer and engine. The signals were simulated using the closed loop non-linear model of aircraft B747 from its associated software package *FTLAB747*.

The simulated signal was pre-processed to enhance the effect of change due to fault. Two types of signatures were created and a clustering technique based on distance from the signatures were used to classify and identify these indicators as normal and faulty conditions. The technique used is based on Singular Value Decomposition*(SVD)* and vector subspace signature method, where the *SVD* is performed on training normal and faulty indicator matrices and the principal singular values are calculated. The corresponding column vectors of matrix V form the two subspaces pertaining to normal condition and faulty conditions.

The new sensor data (indicator) is then classified as normal or abnormal depending on their distance to vector sub-spaces. The sub-space that is closest to the indicator identifies the condition of the indicator. The results showed that more than 85% were classified correctly. This test was carried on for various magnitudes of fault and the results were found to be consistent. The online fault detection algorithm was also implemented, and the results show that the mean detection delay was less than 8 seconds. Also it was showed that the fault detection algorithm performs well for high intensity faults and for faults occurring before the

plane becomes stable. An attempt for fault isolation has been made but with little success. This approach to fault detection could also be applied to other systems, such as power plants, refineries or in the sense to any plant.

These results prove that time-frequency analysis is a promising tool for fault detection. Further research on this method can be done by applying this method effectively for fault isolation. And also this method can be tested for multiple faults. Another interesting future work is the updating of the training sets when a new experiment data arrives.

References

[1] https://ewhdbks.mugu.navy.mil/wavelet.htm.

[2] http://www.sic.rma.ac.be/∼xne/el401/pdf/proj.pdf.

[3] Farzin Aghdasi and Rabab K. Ward. Reduction of boundary artifacts in image restoration. *IEEE Transaction on Image Processing*, 5:611–618, April 1996.

[4] A. Al-Rawi and M. Devaney. Wavelets and power system transient analysis. *Proceedings of IEEE Instrumentation and Measure-ment Technology Conference*. St. Paul, MN, U.S.A.

[5] J. L. Aravena and F. N. Chowdhury. A new approach to fast fault detection in power systems. *Proceedings of the International Conference on Intelligent System Applications to Power Systems*. Orlando, FL, U.S.A.

[6] J. L. Aravena and F. N. Chowdhury. Fault detection of flight critical systems. *Proc. 20th Digital Avionics Systems Conference*, October 2001.

[7] B.R. Bakshi and G. Stephanopoulos. Compression of chemical process data by functional approximation and feature extraction. *AIChE Journal*, pages 477–492, 1996.

[8] P. Balle and R. Isermann. Fault detection and isolation for nonlinear processes based on local linear fuzzy models and parameter estimation. *Proc. 1998 American Control Conf.* Philadelphia, PA.

[9] Marty Brenner and Dale Groutage. Nonstationary dynamics data analysis with wavelet-svd filtering. April 2001. Bozeman,MT, U.S.A.

[10] B. A. Souza Brito, N. S. D. and F. A. C. Pires. Daubechies wavelets in quality of electrical power. *The 1998 International Conference on Harmonics and Quality of Power*. Athens,Greece.

[11] Pallavi Chetan. Blind fault detection using spectral signatures. M.S Thesis, Louisiana State University.

[12] Leon Cohen. *Time-Frequency Analysis*. Prentice-Hall, Upper Saddle River, New Jersey, 1995.

[13] Ingrid Daubechies. Ten lectures on wavelets. *Regional Conference Series in Applied Mathematics, SIAM*. Philadelphia.

[14] L. Desborough and T. Harris. Performance assessment measures for univariate feedback control. *Can. J.Chem. Eng.*, pages 1186–1197, 1992.

[15] Andres Marcos Esteban and Gary J. Balas. A boeing747-100/200 aircraft fault tolerant and fault diagnostic benchmark, June 2003. Aerospace Engineering and Mechanics Department, University of Minnesota.

[16] Luisa Mico Francisco Moreno-Seco and Jose Oncina. A new classification rule based on nearest neighbour search. *Proceedings of the 17th International Conference on Pattern Recognition (ICPR04)*, 2004.

[17] A. W. Galli and G. T. Heydt. Comments on power quality assessment using wavelets. *The 27 Annual NAPS*. Bozeman,MT, U.S.A.

[18] A. Grossman and J.Morlet. Decomposition of hardy functions into square integrable wavelets of constant shape. *SIAM J.Math.Anal.*, 15(4).

[19] C. Hsieh Huang, S. and C. Huang. Application of morlet wavelets to supervise power system disturbances. *The 1997 IEEE/PES Winter Meeting*. New York, NY, U.S.A.

[20] Barbara B. Hubbard. *The world according to wavelets*. A.K.Peters, Wellesley,MA, 1998.

[21] I.T.Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

[22] H. Iyama and J.Kuwamura. Application of wavelets to analysis and simulation of earthquake motions. *Earthquake Engineering and Structural Dynamics*, 28(3).

[23] K.A. Kosanovich and M.J. Piovoso. Pca of wavelet transformed process data for monitoring. *Intelligent data analysis Journal*, 1997.

[24] Storer R.H Ku, W.F. and C. Georgakis. Disturbance detection and isolation by principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, pages 179–196, 1995.

[25] E.L.Russell L.H.Chiang and R.D Braatz. *Fault Detection and Diagnosis in Industrial Systems*. Springer, 2001.

[26] B. Liu and S.F.Ling. On the selection of informative wavelets for machinery diagnosis. *Mechanical systems and signal processing*, 13(1).

[27] Min Luo and J. L. Aravena. Pseudo power signatures for aircraft fault detection and identification. *Proc. 21th Digital Avionics Systems Conference*, October 2002.

[28] Zhong Ren Marius Schmidt, Sudarshan Rajagopal and Keith Moffatyz. Application of singular value decomposition to the analysis of time-resolved macromolecular x-ray data. *Biophysical Journal*, 84:2112–2129, March 2002.

[29] Fabian Morchen. Time series feature extraction for data mining using dwt and dft. October 2003.

[30] Douglas J. Nelson and W. Wysocki. Cross spectral methods with an application to speech processing. *SPIE proceedings of advanced signal processing algorithms, architectures, and implementations IX*, 3807.

[31] P. Nomikos and J.F Macgregor. Multivariate spc charts for monitoring batch processes. *Technometrics*, pages 37–41, 1995.

[32] F. Dovis P.Benotto C.Calosso Olmo, G. and P.Passaro. Instrument-independent analysis of music by means of the continuous wavelet transform. *SPIE proceeding of wavelet applications in signal and image processing VII*, 3813.

[33] Krzysztof Patan. Fault detection of actuators using dynamic neural networks. *2nd Damadics Workshop on Neural Methods for Modelling and Fault Diagnosis*. Genova, Italy.

[34] R.K. Pearson and B.A. Ogunnaike. Detection of unmodelled disturbance effects by coherence analysis. *Proceedings ADCHEM'94*, May 1994.

[35] Teukolsky Saul A. Vetterling William T. Press, William H. and Brian P. Flannery. Numerical recipes in fortran 77: The art of scientific computing. 1992. Cambridge University Press.

[36] M.B. Priestly. *Nonlinear and nonstationary time series analysis*. Academic Press, London, 1988.

[37] Shie Qian and Dapang Chen. *Joint Time-Frequency Analysis*. Prentice-Hall, Upper Saddle River, New Jersey, 1996.

[38] Christos Faloutsos Rakesh Agrawal and Arun N. Swami. Efficient similarity search in sequence databases. *Proceedings of the 4th International Conference of Foundations of Data Organization and Algorithms (FODO)*, pages 69–84, 1993. Chicago, Illinois.

[39] Oliviour Rioul and Martin Vetterli. Wavelets and signal processing. *IEEE SP magazine*.

[40] Camps O. I. Robertson, D. C and J. S. Mayer. Wavelets and power system transients: feature detection and classification. *The SPIE International Symposium on Optical Engineering in Aerospace Sensing*, 2242.

[41] W.J. Staszewski. Structural and mechanical damage detection using wavelets. *Shock and Vibration Digest*, 30(6).

[42] M.L.Scheuer R.J.Sclabassi Sun, Mingui. Decomposition of biomedical signals for enhancement of their time-frequency distributions. *Journal of Franklin Institute*, 337.

[43] Divyakant Agrawal Yi-Leh Wu and Amr El Abbadi. A comparison of dft and dwt based similarity search in time-series databases. *Conference on Information and Knowledge Management (CIKM)*, pages 488–495, 2000.

Appendix A
User Manual

Startup

- Create a directory where to put the training and test data files as *.mat file (i.e. fault_analysis).
- Load and start Matlab v7.0.
- Change the directory to fault_analysis.
- Type at matlab prompt >> multiple.

Selection of Sensor Data

This is the first Graphical User Interface (GUI), that will appear on the screen.

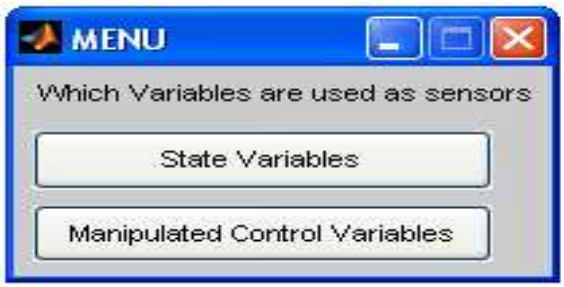


Figure 1

The first question it asks you is *which variables are used as sensors* for this particular analysis. You have two options, State Variables and the Manipulated Control Variables. If you select the state variables as sensors, then in the training phase you have two options whether to use the pre-defined trained subspaces or to run a separate training phase with the input of training data by the user. On the other hand, is you select the manipulated control variables as sensors, then you need to specify the training data later to form the trained subspaces.

Selection of Window Length

Irrespective of the selection of the sensor data, the next GUI that appears is the *selection of the window length*. You have the option of selecting a particular window length for online fault detection among 12, 14, 16, 18 or 20 seconds.

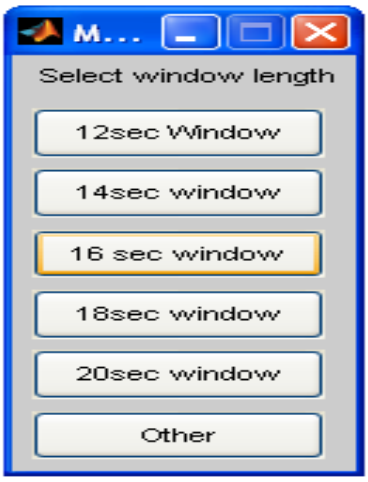


Figure 2

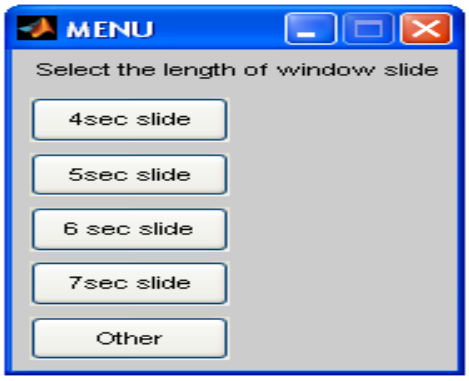
A minimum of 12 seconds is given to provide a clue that the results obtained using a still smaller window showed poor performance. However, for a curious user who wants to analyze for smaller window size, there is an option to enter any window length of his choice by pressing the 'other' button. The following message is displayed

```
Enter the length of window to be selected:
```

You need to specify the window length in seconds.

Selection of window slide

The next GUI asks for the *selection of the window slide*.



The options available are 4, 5, 6 and 7 seconds. Do not select a window slide less than 4 seconds. This increases the computational complexity leading to an error message "Out of memory.Type help" by Matlab. By pressing the 'other' button you will be asked

```
Enter the length of window slide:
```

Again you need to specify the window slide in seconds.

Training Phase

Once the initialization is done, you have the training routine to be executed. As discussed earlier it has different options for the two sensor data: State Variables & Manipulated Control Variables. Initially we shall discuss the selection of state variables as sensor data.

The following GUI appears:

Figure 4

By pressing one of the buttons the program will execute the corresponding subroutine. If you select the 'Use Pre-defined Training Set' option, then the program loads the trained subspace data into memory. Then it asks you to make a selection of the fault you wish to analyze.

Figure 5

At present the program does not support the fault isolation procedure. By selecting a particular fault, the program loads all the files corresponding to that particular fault. The table below shows the type of fault along with their file masks.

Fault	File mask
Elevator	MCU_1
Aileron	MCU_2
Rudder	MCU_3
Stabilizer	MCU_4
Engine	MCU_5
Normal	MCU_0

For example a data by file name MCU_1_0.30462_31.5091_SSdata_1SL represents an elevator failure.

If you select the 'Form a separate training set' button in figure 4 the following note is displayed at the command window.

```
Important Note: Both training normal and fault should have same initial
conditions to obtain better results for fault detection

************************************************************************
Suggestion: For training faulty data select a data with complete fault
and a fault percentage of 20%
```

Figure 6

The above note provides with useful information for selection of training normal and faulty data. Both the data should have same initial conditions and the training faulty data should be selected in such a way that the fault is complete (i.e. starts at 0 seconds) and a loss of efficiency of 20% is observed to be ideal. The program at present does not support forming training subspaces with a database of training data. Next the program displays a dialog box to select the *file for training normal data* as shown above. This displays all the *.mat files that are present in the current directory, and appropriate training normal data is selected.

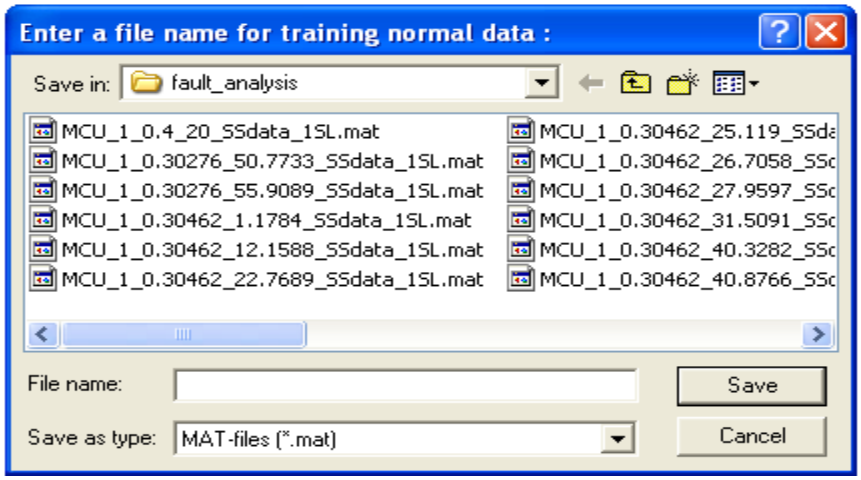


Figure 7

Once this is done, another similar dialog box (figure 8) will ask you for the training faulty data.

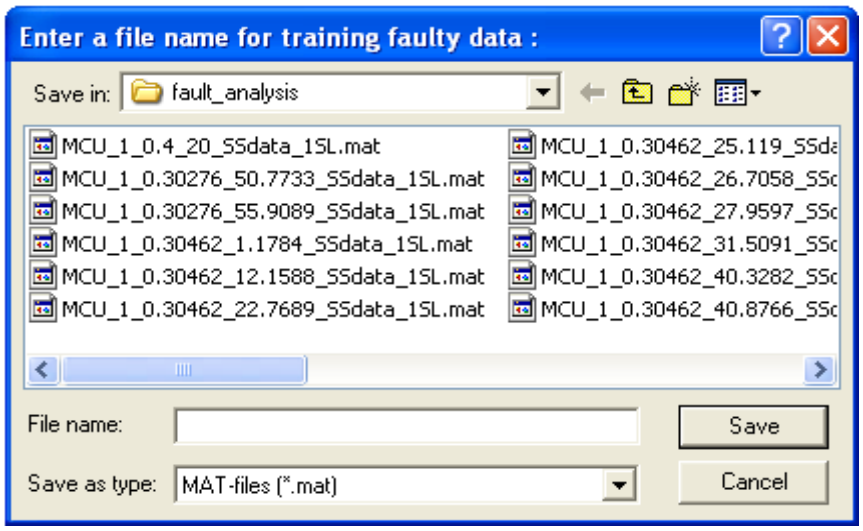


Figure 8

Now the program displays a wait bar as shown below. This routine takes a long time depending on the window length and window slide.

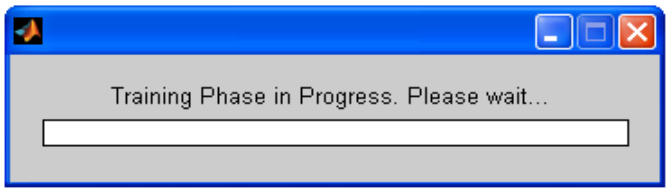


Figure 9

Until now we discussed the training routine if the state variables are selected as sensor data. What if you select the manipulated control variables as sensor data? The program does not have pre-defined trained subspaces and hence follows the routine similar to the one by selecting the 'Form a separate training set' with state variables as sensors. The same dialog boxes appear and the training routine takes a long time to complete.

Classification Phase

Once the training routine is completed, the classification routine starts. The program asks for the directory where all the test files are present (figure 10)

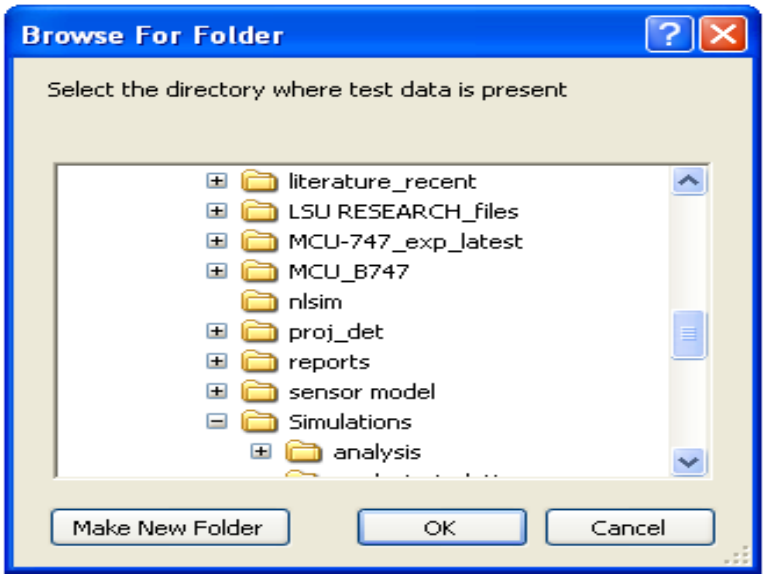


Figure 10

At the startup, we loaded all the test files into the current directory and hence it should be selected. This makes the program less complex. Then the fault detection process starts and displays the following wait bar indicating the progress of the fault detection.

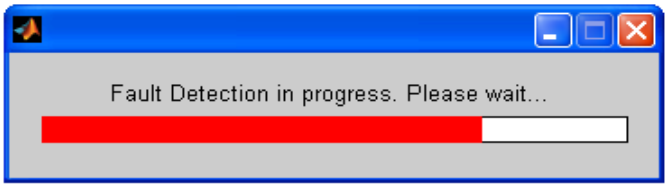


Figure 11

After the fault detection for all the test data is complete the program displays *Analysis Complete* and asks whether to plot the analysis graphs.

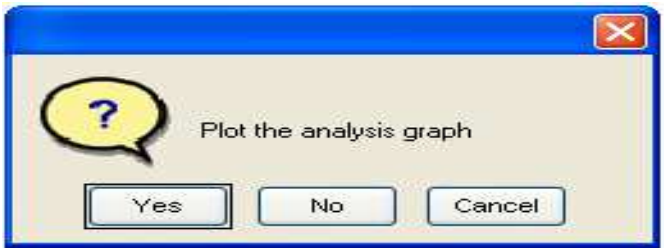


Figure 12

If you press the ‘Yes’ button it displays plots for analysis.

Appendix B
Matlab Programs

Program Listing 1
To perform fault diagnosis for a set of test data

```
%multiple.m
% This routine asks the user the test data and performs the fault diagnosis for every data % and outputs the results in a diary file. Initially the routine trains the systems for a  particular type of fault and based on the trained data classifies the test data as normal or faulty and also detects the onset of fault.

clear
clc
variable=menu('Which Variables are used as sensors','State Variables','Manipulated Control Variables');
% asks user to input which variables are used as sensors
window=menu('Select window length','12sec Window','14sec window','16 sec window','18sec window','20sec window','Other');
% asks user to input the window length for detection. Length of window should be input in seconds.
if window= =1
   winlen=12;
elseif window= =2
   winlen=14;
elseif window= =3
   winlen=16;
elseif window= =4
   winlen=18;
elseif window= =5
   winlen=20;
elseif window= =6
   winlen=input('Enter the length of window to be selected');
end

slide=menu('Select the length of window slide','4sec slide','5sec slide','6 sec slide','7sec slide','Other');
% asks user to input the length of window slide.length of window slide should be input in seconds.
if slide= =1
   slide_len=4;
elseif slide= =2
   slide_len=5;
elseif slide= =3
   slide_len=6;
elseif slide= =4
   slide_len=7;
elseif slide= =5
   slide_len=input('Enter the length of window slide:');
end

winlen=winlen*50;
slide_len=slide_len*50;
```

```
% TRAINING PHASE
if variable= =1
   training=menu('Training Phase','Use Pre-defined Training Set','Form a Seperate Training Set');
   if training= =1
      actuator=menu('Select fault for analysis','Elevator failure','Aileron failure','Rudder failure','Stabilizer
failure','Engine failure','Isolation');
      if actuator= =1
         mask_test='MCU_1';
         name='Elevator';
      elseif actuator= =2
         mask_test='MCU_2';
         name='Aileron';
      elseif actuator= =3
         mask_test='MCU_3';
         name='Rudder';
      elseif actuator= =4
         mask_test='MCU_4';
         name='Stabilizer';
      elseif actuator= =5
         mask_test='MCU_5';
         name='Engine';
      elseif actuator= =6
         mask
      end
   elseif training==2
   disp('Important Note: Both training normal and fault should have same initial')
      disp('conditions to obtain better results for fault detection');
      fprintf('\n')
      disp('*****************************************************************');
      disp('Suggestion: For training faulty data select a data with complete fault')
      disp('and a fault percentage of 20%');
      pause(3)
      fprintf('\n')
[mask_normal,norm_path]=uiputfile('*.mat','Enter a file name for training normal data :');
[mask_faulty,fault_path]=uiputfile('*.mat','Enter a file name for training faulty data :') ;
      training_xobs(winlen,slide_len,mask_normal,mask_faulty)
   end
elseif variable==2
      disp('Important Note: Both training normal and fault should have same initial')
      disp('conditions to obtain better results for fault detection');
      fprintf('\n')
      disp('*****************************************************************');
      disp('Suggestion: For training faulty data select a data with complete fault')
      disp('and a fault percentage of 20%');
      pause(3)
      fprintf('\n')
[mask_normal,norm_path]=uiputfile('*.mat','Enter a file name for training normal data :');
[mask_faulty,fault_path]=uiputfile('*.mat','Enter a file name for training faulty data :') ;
      h1 = waitbar(0,'Training Phase in Progress. Please wait...');
      training_u(winlen,slide_len,mask_normal,mask_faulty)
      close(h1)
```

```
end
d_test=uigetdir('','Select the directory where test data is present');
if variable==1 & training==1
   test_data=d_mcu(d_test,mask_test,'.mat'); % list all the files with the given mask in the mentioned directory
elseif variable==1 & training==2
   test_data=d_mcu(d_test,mask_faulty(1:5),'.mat'); % list all the files with the given mask in the mentioned directory
elseif variable==2
   test_data=d_mcu(d_test,mask_faulty(1:5),'.mat'); % list all the files with the given mask in the mentioned directory
end
clc
decision_final=[];
time_fault=[];delay=[];
original_fault_onset=[];
h = waitbar(0,'Fault Detection in progress. Please wait...');

%CLASSIFICATION PHASE
for i=1:length(test_data)          % for loading the test data one after the other
   mcu_testdata=test_data(i).name; % test data for present iteration
   if variable==1

[decision,fault_onset_orig,fault_intensity]=single_testdata_window_xobs(mcu_testdata,winlen,slide_len,actuator);    % analysis for the present test data
   elseif variable==2
      [decision,fault_onset_orig,fault_intensity]=single_testdata_window_u(mcu_testdata,winlen,slide_len);
% analysis for the present test data
   end
   original_fault_onset(i)=fault_onset_orig;      %saving the original fault onset time
   intensity(i)=fault_intensity*100;              %saving the intensity of fault
   if length(find(decision==1))~=0
      for j=1:length(decision)
         if decision(j)==1
            time_fault(i)=((winlen+(j-1)*slide_len))/50;       % converting the indicator index to detected fault time
            break
         end
      end
      delay(i)=time_fault(i)-original_fault_onset(i);     % display detection delay
      fprintf('\n')
      disp(['Analysis of fault detection for ',mcu_testdata])
      disp('----------------------------------')
      disp(['Onset of fault : ',int2str(original_fault_onset(i)),'sec'])
      disp(['Fault detected at : ',int2str(time_fault(i)),'sec'])
      disp(['Detection delay  : ',int2str(delay(i)),'sec'])
      clc
   elseif length(find(decision==1))==0
       delay(i)=60;
       disp(['Analysis of fault detection for ',mcu_testdata])
       disp('----------------------------------')
```

```
            fprintf('\n')
            disp('Missed Alarm')
            clc
        end
        decision_final(i,:)=decision;
        waitbar(i/4)%length(test_data)

    end
    close(h)
    disp('Analysis complete');
    button=questdlg('Plot the analysis graph');
    if strcmp(lower(button),'yes')
    plot(intensity,original_fault_onset,delay);
    end
```

Program Listing 2

Calls the training phase for each sensor to obtain the training signature subspaces

```
% This function performs the training phase for each sensor and returns the V matrices along with the best scales.

function training_xobs(winlen,slide_len,mask_normal,mask_faulty)
   [v_normal_4,v_faulty_4,a_4]=training_refined_xobs(4,winlen,slide_len,mask_normal,mask_faulty);
[v_normal_5,v_faulty_5,a_5]=training_refined_xobs(5,winlen,slide_len,mask_normal,mask_faulty);
[v_normal_6,v_faulty_6,a_6]=training_refined_xobs(6,winlen,slide_len,mask_normal,mask_faulty);
[v_normal_7,v_faulty_7,a_7]=training_refined_xobs(7,winlen,slide_len,mask_normal,mask_faulty);
[v_normal_8,v_faulty_8,a_8]=training_refined_xobs(8,winlen,slide_len,mask_normal,mask_faulty);
[v_normal_9,v_faulty_9,a_9]=training_refined_xobs(9,winlen,slide_len,mask_normal,mask_faulty);
[v_normal_10,v_faulty_10,a_10]=training_refined_xobs(10,winlen,slide_len,mask_normal,mask_faulty);

 save data_training
```

Program Listing 3

Determines the normal and faulty subspaces for training data of each sensor.

```
% This function determines the normal and faulty subspaces for training data of each state.
% This function receives the state to detect the subspaces using the traning data already existing.
% Use [v1,v2,a]=training_refined_xobs(state,winlen,slide_len,mask_normal,mask_faulty)
% Inputs:
%     'state' to specify the state of the training data to find subspaces
%     'winlen' is the window length for analysis
%     'slide_len' is the length of window slide
%     'mask_normal' is the training normal data
%     'mask_faulty' is the training faulty data
% Outputs:
%       'vlpr_dat' is the desired v-matrix respresenting the normal training subspace
%       'vlpo_dat' is the desired v-matrix respresenting the faulty training subspace
%       'a' specifies the scales for the cwt.
```

```
function [v1pr_dat,v1po_dat,a]=training_refined_xobs(state,winlen,slide_len,mask_normal,mask_faulty)

    y1=[];y2=[];
 %compute the fft and plot the magnitude plot to analyze the energy distribution in different frequencies of normal data.

    bN=pow2(nextpow2(length(sd1norm))); %points for the FFT
    load(mask_normal);          %load normal data which is the training data
    sd1norm=xobs(:,state);
    bN=pow2(nextpow2(length(sd1norm))); %points for the FFT
    y1 = fft(sd1norm',bN );
    m1 = abs(y1);

 %compute the fft and plot the magnitude plot to analyze the energy distribution in different frequencies of faulty data.

    load(mask_faulty);
    sd1fault=xobs(:,state);
    y2=fft(sd1fault',bN);
    m2=abs(y2);
    m=abs(m1-m2); % we don't care about the signs
        fs = 50; %sampling frequency
    f=(fs/bN)*[0:bN/2-1]'; %this is actual frequency in Hz. Max is fs/2

    % selection of peaks and corresponding frequencies from the above frequency difference plot
    [msort, isort]=sort(m(1:bN/2)); %sort ascending the abs (m1-m2)
    idown=flipud(isort(:)); %reverse order to descending
    fbest=f(idown(1:17)); %Best 17 frequencies. One more than needed beacuse f=0 is in the list
    fc=centfrq('dB4'); %this is center frequency in Hz!!
    fselect=sort(fbest); %sort frequencies ascending
    fselect=fselect(2:17); % I know that f=0 is in the list. Cannot have a=infty
    a=fc./fselect;

    load(mask_normal);          %load normal data which is the training data

    sd1post=[];
    for w=1:slide_len:(size(xobs,1)-winlen)
        cwt_sd1post=abs(cwt(xobs(w:w+winlen-1,state),a,'dB4'));
        sd1post=[sd1post cwt_sd1post];
    end
      sd1norm_sens=normc(sd1post);

    load(mask_faulty)

    sd1fault=[];
    for w=1:slide_len:(size(xobs,1)-winlen)
        cwt_sd1fault=abs(cwt(xobs(w:w+winlen-1,state),a,'dB4'));
        sd1fault=[sd1fault cwt_sd1fault];
    end
        sd1fault_sens=normc(sd1fault);
```

```
%Formation of a 3D subspaces for both total normal and total faulty data using only the sensitive scales.
[u1pr_dat,s1pr_dat,v1pr_dat]=svd(sd1norm_sens');
diag1=diag(s1pr_dat);

[u1po_dat,s1po_dat,v1po_dat]=svd(sd1fault_sens');
diag2=diag(s1po_dat);
```

Program Listing 4

To run fault detection analysis for each window of the data.

```
% single_testdata_window_xobs.m
% This function determines the condition for all windows of the data
% This function receives the testdata,window length,length of window slide and the type of fault for which analysis to be done.
% Use [decision_window,time1,fp]=single_testdata_window_xobs(data,winlen,slide_len,actuator)
% Inputs:
%     'data' inputs the data of sensor for the validation data.
%     'winlen' is the window length for analysis
%     'slide_len' is the length of window slide
%     'actuator' to specify the type of fault under consideration
% Outputs:
%       'decision_window' gives a decision (faulty=1,normal=0) for all windows considered
%       'time1' returns the original fault onset time
%       'fp' returns the fault intensity of  the data
function [decision_window,time1,fp]=single_testdata_window_xobs(data,winlen,slide_len,actuator)

dist_normal=[];dist_faulty=[];c=1;
% Load training data depending on the type of fault
if actuator= =1
   load training_newdata_elevator_600_200
   time1=MCU_column_time1;
   fp=MCU_column_fp;
elseif actuator==2
   load training_data_aileron
   time1=MCU_wheel_time1;
   fp=MCU_wheel_fp;
elseif actuator==3
   load training_data_rudder_1000_500
   time1=MCU_pedal_time1;
   fp=MCU_pedal_fp;
elseif actuator==4
   load training_data_stabilizer
   time1=MCU_stab_time1;
   fp=MCU_stab_fp;
elseif actuator==5
   load training_data_engine
   time1=MCU_EPR1_time1;
   fp=MCU_EPR1_fp;
end
```

```
load(data);
    testdata=xobs;
    [rr,cc]=size(testdata);
    cs=1; % count on states
    decision_window=[];
    count=1;

for w=1:slide_len:rr-winlen
    xx=testdata(w:w+winlen-1,1:10);
    for s=4:10      % consider only the states sensitive to fault detection
            if s==4
                v_normal=v_normal_4;
                v_faulty=v_faulty_4;
                a=a_4;
            elseif s==5
                v_normal=v_normal_5;
                v_faulty=v_faulty_5;
                a=a_5;
            elseif s==6
                v_normal=v_normal_6;
                v_faulty=v_faulty_6;
                a=a_6;
            elseif s==7
                v_normal=v_normal_7;
                v_faulty=v_faulty_7;
                a=a_7;
            elseif s==8
                v_normal=v_normal_8;
                v_faulty=v_faulty_8;
                a=a_8;
            elseif s==9
                v_normal=v_normal_9;
                v_faulty=v_faulty_9;
                a=a_9;
            elseif s==10
                v_normal=v_normal_10;
                v_faulty=v_faulty_10;
                a=a_10;
            end
       %    Computation of distance clusters
              [d1,d2]=distance(xx,v_normal,v_faulty,s,a);
       dist_normal(cs,:)=d1;       %distance to normal subspace
       dist_faulty(cs,:)=d2;       %distance to faulty subspace
       cs=cs+1;          % increment count on the states
    end

    % Voting Mechanism and decision making.

    sensor=[];
    decision=[];
    for i=1:length(dist_normal)
```

```
    for j=1:size(dist_normal(:,1))
        if dist_normal(j,i)>dist_faulty(j,i)
            sensor(j,i)=1;
        elseif dist_normal(j,i)<dist_faulty(j,i)
            sensor(j,i)=0;
        end
    end
  end
  for i=1:length(dist_normal)
    if length(find(sensor(:,i)==0))>length(find(sensor(:,i)==1))
        decision(i)=0;
    elseif length(find(sensor(:,i)==0))<length(find(sensor(:,i)==1))
        decision(i)=1;
    else
        decision(i)=2;
    end
 end
  c=0;d=0;
   for i=1:length(dist_normal)
      if decision(i)==0
          c=c+1;
      elseif decision(i)==1
          d=d+1;
      end
   end
   if c>d
      decision_window(count)=0;
   elseif c<d
      decision_window(count)=1;
   end
   count=count+1;
end
```

Program Listing 5

To run classification phase.

```
%distance.m
% This function determines the distance of testdata to normal and faulty training subspaces.
% This function receives the testdata,subspace v-matrices,state and the scales for computing cwt.
% Use [distance1,distance2]=distance(test_data,v1pr_dat,v1po_dat,state,a)
% Inputs:
%     'test_data' inputs the data of state variables for the validation data.
%     'vlpr_dat' is the v-matrix respresenting the normal subspace
%     'vlpo_dat' is the v-matrix respresenting the faulty subspace
%     'state' to specify the state of the training data to find subspaces
%     'a' specifies the scales for the cwt.
% Outputs:
%       'distance1' is the distance clusters to normal subspace
%       'distance2' is the distance clusters to faulty subspace
```

```
function [distance1,distance2]=distance(test_data,v1pr_dat,v1po_dat,state,a)

   xob=test_data(:,state);
   %xob=[zeros(249,1);xob];     %zero padding the signal
   [rr,cc]=size(xob);
   xx=[flipud(xob);xob;flipud(xob)];      %Mirroring on both sides to remove edge effect
   sd1post_dat=abs(cwt(xx,a,'dB4'));      %CWT for the 9000 data samples
   sd1post_dat=sd1post_dat(:,length(xob):2*length(xob)-2);     %Consider only the middle 3000 data samples
which are required for analysis

   % CALCULATION OF DISTANCE CLUSTERS TO NORMAL AND FAULTY SUBSPACES
   alpha1_dat=[];    alpha2_dat=[];
   sd1post_dat_fin=normc(sd1post_dat);
   for x=1:8
      alpha1_dat(x,:)=dot(sd1post_dat_fin,(v1pr_dat(:,x)*ones(1,size(sd1post_dat_fin',1))));
      alpha2_dat(x,:)=dot(sd1post_dat_fin,(v1po_dat(:,x)*ones(1,size(sd1post_dat_fin',1))));
   end
   shat1_dat=zeros(size(sd1post_dat_fin));shat2_dat=zeros(size(sd1post_dat_fin));
   for y=1:8
      shat1_dat=shat1_dat+(v1pr_dat(:,y)*alpha1_dat(y,:));
      shat2_dat=shat2_dat+(v1po_dat(:,y)*alpha2_dat(y,:));
   end

   dif1_dat=zeros(size(sd1post_dat_fin));
   dif2_dat=zeros(size(sd1post_dat_fin));
   dif1_dat=sd1post_dat_fin-shat1_dat;
   dif2_dat=sd1post_dat_fin-shat2_dat;
   dist1_dat=zeros(size(sd1post_dat_fin));
   dist2_dat=zeros(size(sd1post_dat_fin));
   dist1_dat=sqrt(sum(dif1_dat.*dif1_dat));
   dist2_dat=sqrt(sum(dif2_dat.*dif2_dat));

   distance1=dist1_dat;      %distance to normal subspace
   distance2=dist2_dat;      %distance to faulty subspace
```

Program Listing 6

To obtain a list of files present in a directory

```
% d_mcu.m
%This script receives the name of a directory, a mask and returns
% a list of names satisfying a given mask
% USE mcu = d_mcu(d, mask, d_ext)
% Inputs:
% 'd' is the string with path to directory (full path if not in current directory)
% 'mask' is the  mask to identify required files. No wild card characters
% 'd_ext' is the desired extension (INCLUDING THE PERIOD!)
% Outputs:
% 'mcu'  outputs structure following dir command in MATLAB(name, date, bytes, isdir)
function mcu=d_mcu(d, mask, d_ext)
```

```
mcu=struct([]);
allf=dir(d); % lists everything in that directory
nf=size(allf,1); %this many files
if ~nf
    disp(['no files found in ' d])
    return
else
    kf=0; % no files yet
    for k=1:nf
        flk=allf(k).name; %this is the name
        [f_path,f_name, f_ext,f_ver]=fileparts(flk);
        if strcmp(lower(f_ext),d_ext) & ~isempty(strfind(flk,mask))
%found a MAT file with the right mask
            kf=kf+1;
            mcu=[mcu;allf(k)];
        end
    end
end
```

Vita

Venkata S Vongala was born on March 10th, 1981, in Vijayawada City, Andhra Pradesh, India. He finished his undergraduate studies at Andhra University April 2003. In August 2003 he came to Louisiana State University to pursue graduate studies in electrical engineering. He is currently a candidate for the degree of Master of Science in Electrical Engineering, which will be awarded in December 2005.